October 2019

A Study on Controlling Power Supply Ramp-Up Time in SRAM PUFs

Harshavardhan Ramanna

A STUDY ON CONTROLLING POWER SUPPLY RAMP-UP TIME IN SRAM PUFS

A Thesis Presented

by

HARSHAVARDHAN RAMANNA

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL AND COMPUTER ENGINEERING

September 2019

Electrical and Computer Engineering

A STUDY ON CONTROLLING POWER SUPPLY RAMP-UP TIME IN SRAM PUFS

A Thesis Presented

by

HARSHAVARDHAN RAMANNA

Approved as to style and content by:

Daniel Holcomb, Chair

Maciej J. Ciesielski, Member

Wayne Burleson, Member

Robert W. Jackson, Department Chair
Electrical and Computer Engineering

DEDICATION

To my parents.

ACKNOWLEDGMENTS

This thesis would not have been possible without the support and advice of Professor Daniel Holcomb. His expertise and patience in guiding the research has had a profound impact on my work. I am grateful for his useful insights and encouragement throughout my graduate program and would like to take this opportunity to thank him for his support.

I am also indebted to the professors of the Electrical and Computer Engineering department at the University of Massachusetts, Amherst for their selfless dedication in educating and inspiring me during my graduate studies. The knowledge and skills imparted by them during my program has proven useful not just in research but also outside of the academic world.

I am also appreciative of the amazing opportunity to be a part of the Hardware Security Group of the ECE Department. Very few are fortunate enough to be surrounded by talented people of such high calibre. I want to acknowledge the help that my colleague Siva Dhanuskodi has provided throughout my thesis research work. He has been closer to a mentor than a colleague.

Last but not the least, I want to thank my family and friends for their unwavering support.

ABSTRACT

A STUDY ON CONTROLLING POWER SUPPLY RAMP-UP TIME IN SRAM PUFS

SEPTEMBER 2019

HARSHAVARDHAN RAMANNA

B.Tech., NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA

M.S.E.C.E., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Daniel Holcomb

With growing connectivity in the modern era, the risk of encrypted data stored in hardware being exposed to third-party adversaries is higher than ever. The security of encrypted data depends on the secrecy of the stored key. Conventional methods of storing keys in Non-Volatile Memory have been shown to be susceptible to physical attacks. Physically Unclonable Functions provide a unique alternative to conventional key storage. SRAM PUFs utilize inherent process variation caused during manufacturing to derive secret keys from the power-up values of SRAM memory cells.

This thesis analyzes the effect of supply ramp-up times on the reliability of SRAM PUFs. We use SPICE simulations as the platform to observe the effect of supply ramp times at the circuit level using carefully controlled supply voltages during power-up. We also measure the effect of supply ramp times on commercially available SRAM ICs by performing reliability and uniqueness measurements on two commercial

SRAM models. Finally, a hardware implementation is proposed in a commercial 16nm FinFET technology to establish the design flow for taping out a custom SRAM IC with separated peripheral and core power supplies that would allow for experimental evaluation of sequenced power supplies on the SRAM PUF.

TABLE OF CONTENTS

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

CHAPTER 1

INTRODUCTION

The field of security has grown both in importance as well as complexity over the years. With increase in exposure to third parties, protecting encrypted data is more relevant than ever. Due to the ready availability of computing power combined with improved methods of attacking, it is necessary to utilize more robust measures to ensure security of secret keys and thus the data stored in hardware. Key storage in Non-Volatile Memory such as Flash has proven to be susceptible to physical attacks [45] [3]. This has paved the way to Physically Unclonable Functions (PUFs) [19] which offer an innovative alternative to classical cryptography. PUFs can be considered as a type of hardware entangled secret that exploits inherent physical features that are unique to the underlying hardware modules. They form specialized modules that are easy to evaluate but almost impossible to predict or replicate. PUFs have formed an important facet of hardware security and have proven useful for device authentication, key generation [52] and key exchange [8]. This thesis focuses on one such PUF called the SRAM Power-Up PUF [23] [21] [24].

When a stimulus is applied to the PUF structure, the PUF is expected to behave in a seemingly random or unpredictable way. The stimulus is called a challenge to the PUF. The applied stimulus interacts with the physical micro structure of the device to produce an output that can be measured. The produced output is called a response. PUFs thus, implement a challenge-response type of authentication, where the party requiring authentication must provide a valid response to a challenge that is posed to it. The combination of the input challenge and the output response form a Challenge-

Response Pair (CRP). The advantage of PUFs is that the CRP for a PUF instance is unpredictable but repeatable. PUFs have seen uses in numerous application such as FPGA bitstream IP protection [21] and have also been implemented commercially in FPGAs such as Xilinx Ultrascale [34] for secret key generation.

1.1 Process Variation

The randomness in the PUF responses are caused by the process variation during manufacturing as well as changes in environmental factors [36]. One way of categorizing process variation is into global and local mismatch factors [43]. Global factors are those independent of geometry and placement while local factors encapsulate those dependent on locality.

Another way process variations can be classified are as systemic and random variations [1]. Systemic variations are deterministic and dependent on the structure as well as the topology of the gate or transistor. These are caused by known physical phenomenon during manufacturing. For example, Chemical Mechanical Planarization (CMP) results in different wire thickness across the chip depending on the routing density. Random variations are those which cannot be determined or predicted. The different sources of random variations [15] are:

- **Edge effects** - As lithography involves multiple masks and as the lengths of the features reduce below the wavelength of light, edges forming the various device layers vary between transistors. This is the main source of process variation for FinFETs.

- **Random dopant fluctuations** - Differences in the implanted dopant concentration can alter the properties of the transistor such as threshold voltages. Due to reduced geometry in advanced technology nodes, dopant fluctuations has a

larger effect because the total number of dopants is fewer. This is the main source of variation in MOSFETs.

- **Oxide Effect** - Fluctuations in oxide thickness during manufacturing affect the threshold voltages of the transistors causing mismatch between devices.

Random variations can be further classified as intra-die and inter-die process variation. Inter-die process variation are differences in transistor parameters on different ICs. These feature differences change not just from die to die but also from wafer to wafer as well as wafer lot to wafer lot. Intra-die process variation on the other hand is the variation in transistors present on the same chip. Intra-die process variations exhibit spatial as well as structural correlation. Devices located close to each other tend to have similar features as do similarly structured gates or transistors present on the same die. With reduction in device sizes, both intra-die and random fluctuations have increased drastically [5].

1.2 Physically Unclonable Functions

Several Physically Unclonable Functions have been proposed and implemented. Some of the examples are Anderson PUF [2], the Ring Oscillator PUF [47], Sensor based PUF [13] [55], Bistable PUF [10] [11]. Although all PUFs are based on utilizing physical measurements of a random underlying process, PUFs can be categorized based on several factors as shown in Figure 1.1. If the source of randomness of the PUF is intrinsically present in the physical system, the PUF is categorized as an Intrinsic PUF. If the source of randomness is explicitly introduced into the system, the PUF is termed as an extrinsic PUF. Another way to categorize PUFs is based on the challenge response pairs of the PUFs [42]. If the PUF has a limited set of responses or just a single response, it is categorized as a weak PUF. On the other hand, if the PUF has an exponential CRP mapping space, it is termed as a strong

PUF [41]. Due to the large number of CRPs, strong PUFs prevent an attacker from performing a full read out of all CRPs.

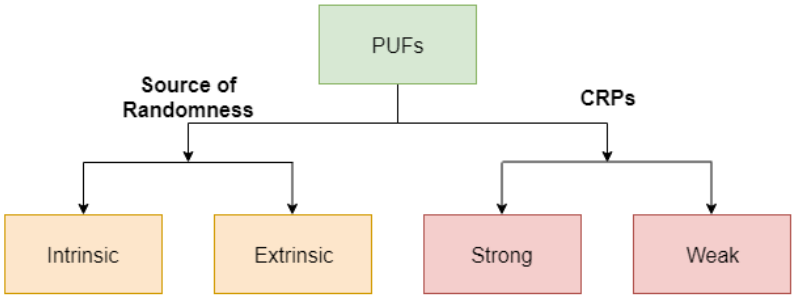


Figure 1.1: A categorizing of PUFs.

An example of an extrinsic PUF is the Optical PUF [37] [40] shown in Figure 1.2. The optical PUF consists of a transparent material that is doped with light scattering particles. When a beam of monochromatic light shines on the coated material, a random and unique speckle pattern will be generated. The placement of the light scattering particles is an uncontrolled process and the interaction between the light and the particles is very complex. Therefore, it is very hard to duplicate the response of the optical PUF such that the same speckle pattern will arise, hence the postulation that it is unclonable.

A via PUF [29] on the other hand is an example of an intrinsic PUF. A via PUF is based on randomly generated via-hole formation during IC fabrication. IC manufacturers establish a design rule regulating the minimum via hole size to guarantee the physical connection between metal layers formed by a via. The via PUF intentionally violates this design rule to create uncertainty in the via connection. The advantages of using via holes are that the open or short states in the circuit create very clear voltage or current differences, and once the open or short states are determined, they are not changed over time by environmental factors and do not require expensive

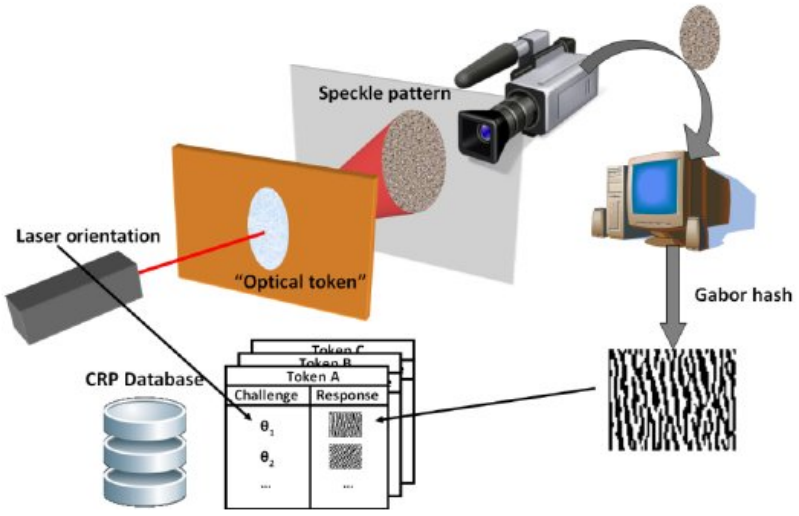


Figure 1.2: The implementation of the optical PUF [32].

post-processing such as error correction. Figure 1.3 shows a cross section of a via PUF showing the different vias acting as short circuit and open circuits in the IC.

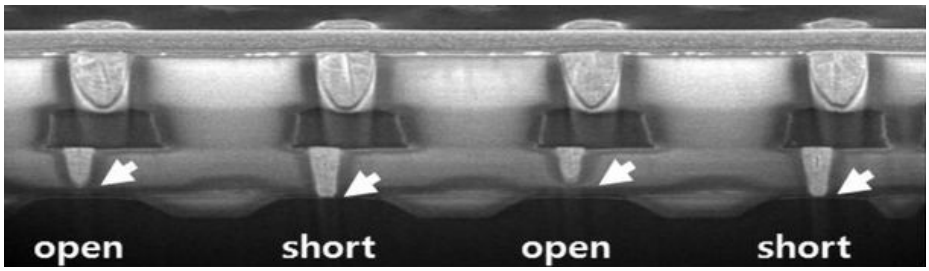


Figure 1.3: A cross section of the via PUF [29].

The Butterfly PUF [30] is an example of a weak PUF. A butterfly PUF consists of two cross coupled latches that are forced into an unstable state before allowing them to settle into a stable state as shown in Figure 1.4. Each latch contains an asynchronous preset and an asynchronous clear pins, and the cross coupled latch system has two stable states. One of the latches is sent a preset signal while the other is sent a clear signal through an excitation signal. This causes the system to enter an unstable state and when the excitation signal is removed, the PUF settles

into one of the two stable states. The state that the PUF settles into depends on the interconnect delays of the two latches. This is a weak PUF as there is only one response per PUF instance.

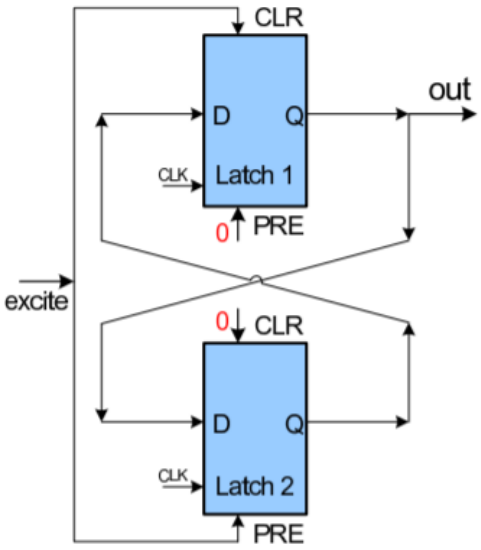


Figure 1.4: Implementation of Butterfly PUF [30].

An example of a strong PUF on the other hand is an Arbiter PUF [9] [44]. An Arbiter PUF implementation depicting the path of signals based on the challenge is shown in 1.5. The Arbiter PUF exploits the difference in the delays of two multiplexer paths. The exact path that each signal races through is determined by k external bits which are applied at the stages, one bit per stage. These k bits form the challenge to the Arbiter PUF. An arbiter decides which signal arrived first or won the race and outputs a 1-bit response. Since the delay between the various multiplexer paths are dependent on process variation, the PUF response depends on the input challenges. Thus the Arbiter PUF has 2^k challenges and 1-bit response for each of those challenges.

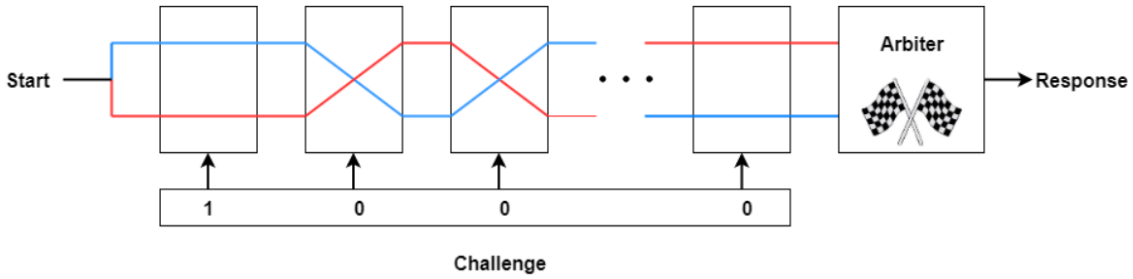


Figure 1.5: Implementation of an Arbiter PUF where a low challenge bit switches the path and a high challenge bit maintains the path in the switcher modules [20].

1.3 Key Generation using PUF

PUF responses are influenced by not just process variation but by noise and environment factors as well. PUF responses are often not uniformly distributed across the PUF. Due to these reasons, PUF responses cannot be directly used as cryptographic keys [7]. To overcome the issue of noise and non-uniformity, Helper Data Algorithms (HDA) [12] are used to generate cryptographically secure keys. HDA ensures that the keys generated are reproducible and have high-entropy. The key generation is split into two phases - enrollment and regeneration.

During the enrollment phase, the "true" response of the given PUF is obtained. Depending on the length of the response R as well as the error rate that needs to be corrected, a code word C_s is selected from an Error Correcting Code C. The response R is given as the input to the HDA to produce the Helper Data HD and a secret key K. Although more advanced key generation methods have been proposed recently [38] [31], we explain key generation using a simple method called code-offset technique [14] shown in Figure 1.6.

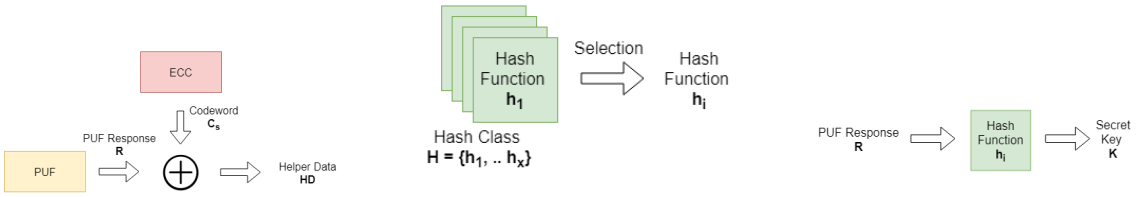
1. First, the bit offset between C_s and R produces the Helper Data HD through equation 1.1

$$HD \leftarrow C_s \oplus R \tag{1.1}$$

2. The secret key is produced through a hash function h_i belonging to the universal hashing function family H. The hash function h_i is selected to ensure maximum entropy in the generated secret key.

$$K \leftarrow h_i(R) \tag{1.2}$$

The Helper Data HD is available publicly and can be stored in the NVM of the chip as long as an attacker cannot manipulate it. Although the Helper Data inevitably leaks information [35] about the PUF response R, the information loss can be offset by utilizing additional bits in the PUF response.



(a) (b) (c)

Figure 1.6: Figure 1.6a shows the generation of Helper Data HD. Figure 1.6c shows the generation of the secret key.

The key regeneration phase shown in Figure 1.7 consists of the following steps:

1. The PUF response R' is collected from the test PUF that needs to be authenticated. By using the Helper Data provided from the enrollment stage HD, a code word C'_s is generated using equation 1.3.

$$C'_s \leftarrow R' \oplus HD \tag{1.3}$$

2. The code word C'_s can be corrected to C_s if the two code words are separated by a small Hamming Distance. Thus, selection of an appropriate ECC algorithm

is imperative in ensuring that there no false negatives in matching the PUF responses from enrollment and regeneration.

$$C_s \leftarrow \mathbf{ECC}(C'_s) \tag{1.4}$$

3. From the reconstructed code word C_s and the Helper Data HD, we can generate the enrolled PUF response R using equation 1.5.

$$R \leftarrow C_s \oplus HD \tag{1.5}$$

4. The reconstructed response R can then be used to generate the secret key K using the hashing function h_i.

$$K \leftarrow h_i(R) \tag{1.6}$$

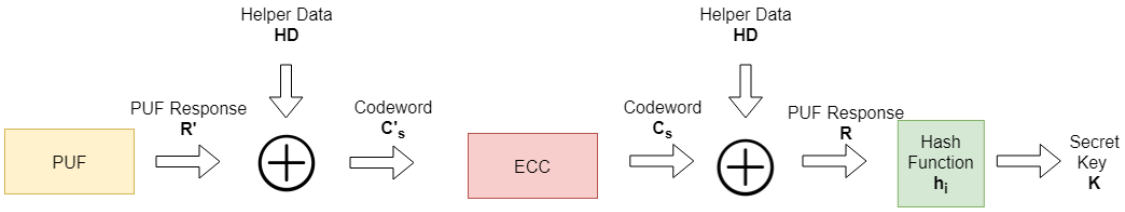


Figure 1.7: Process for secret key regeneration from noisy PUF response.

1.4 Error Correction and Reliability Improvements

As mentioned in the previous section, due to the difference in PUF responses at the time of enrollment and key regeneration, error correction is a vital aspect of PUF based keys. Although various techniques for ECC have been suggested, most of them suffer from having an expensive overhead when being implemented in hardware.

The earliest error correction for PUF based keys suggested the use of 2-D Hamming Codes [18] [56]. The proposed 2-D Hamming Code ECC arranges the PUF response into rows and columns, creating a matrix. Hamming Codes are used to produce redundant information for each row and column of the matrix. This redundant information along with parity bits is used to correct errors in the PUF responses during key regeneration. The main drawback of this method is that the matrix of PUF response created during regeneration cannot have more than one row with two errors. A more appropriate ECC method for PUFs utilizes Bose-Chaudhuri-Hochquenghen (BCH) codes [46]. The first proposed implementation was capable of correcting 30 errors in a PUF response of 255 bits. The proposed implementation exposed 192 bits of Helper Data, thereby requiring the key size to be reduced to 63. Several other methods for improving error correction in PUFs have been suggested. Index-Based-Syndrome (IBS) [56] is shown to leak less information than conventional ECC which use bitwise XOR-masks. Other improvements include soft decision information in the Helper Data Algorithm that reduces the number of PUF response bits required to generate a secure key [31]. Suggestions to reduce ECC complexity include two stage coding through the use of repetition coding and XOR-mask syndrome generation [7].

While the previously mentioned techniques deal with ECC itself, others papers suggest reducing ECC overhead by improving the reliability of PUFs. Helper Data required to generate a secret key increases with the unreliability of the PUF response leading to larger information leak about the PUF response. Thus, to ensure a secure cryptographic key, the PUF response bits need to be increased to overcome the entropy loss. The area overhead required for ECC and the associated increase in the number of PUF response bits to overcome the unreliability grows superlinearly with the error rate [27]. Designers save area overhead when the reliability of the PUF increases. For example, in an FPGA based PUF implementation, reducing the BER from 10% to 3% reduces the ECC LUT and register usage by over 40% [53].

Similarly, ASIC implementations of PUFs show 60% area savings when the error rate drops from 20% to 5% [39].

One way to improve reliability is by utilizing the phenomenon of Negative Bias Temperature Instability (NBTI)[6] [17]. When a MOSFET operates under a negative gate-to-source voltage, positive charges accumulate under the gate leading to an increase in the threshold voltage. NBTI can be artificially achieved at high temperatures by subjecting a PMOS of the SRAM to negative bias. By selectively targeting one of the PMOS in the SRAM cell, the mismatch between the inverters is increased leading to more repeatable power-up responses. Several circuit enhancement techniques have also been proposed to achieve improved reliability. One such method is to replace the 6T transistor in the memory core with a 8T cell that contains an embedded latch which reinforces the bit value during power-up [28]. This method changes the drive strength of the PMOS transistors dynamically to make the bit cells less prone to noise and voltage fluctuations. Other improvements suggest modifying the 6T structure to utilize active resistive loads, parallel loads or current mirror loads [39]. These alterations make the SRAM cell more sensitive to process variations leading to a more reliable PUF construction.

1.5 Hypotheses

Most of the previously discussed methods for improving reliability require changing SRAM cell structure or various methods of post-processing the PUF responses. The ubiquity of SRAM PUFs stems from the ability to use existing SRAM circuitry that are foundry-provided, tested and require no modifications, with post processing circuits. Keeping this in mind, we propose a non-intrusive technique to improve the reliability of SRAM PUFs. The main hypothesis of this research is that a large ramp-up time during power-up of SRAM cells should minimize the effect of noise on the power-up state, thereby improving reliability. This would mean that the Power-Up

SRAM PUF's responses would be more reliable if the power-up ramp times are larger. Another hypothesis is that separating power supplies to the SRAM memory core and its peripheral circuitry can improve reliability of the SRAM PUF by allowing for a controlled power-up sequence that is less influenced by the peripheral circuitry. To test these hypotheses, we propose to perform the following three experiments:

1. Perform circuit simulation of SRAM cells and SRAM modules from a foundry-provided SPICE model file. SPICE simulations help analyze the effect of the ramp-up time at the transistor level to evaluate the hypotheses under ideal conditions with maximum observability. Although it allows for better control over the test environment, it is slow and not scalable.

2. Measure BER of SRAM power-up using commercial SRAM ICs for different ramp-up times and compare it to the results of SPICE simulations. Testing commercially available SRAM ICs allows us to test our hypothesis in a realistic setting and for different technology nodes. The drawback in this setup is that most commercially available ICs provide a single power pin to the SRAM. Thus, we cannot test the hypothesis of improving reliability by separating power supplies to the core and peripheral circuit. We also do not know the implementation details of the ICs.

3. Design and tape-out an SRAM IC with separate power supplies to the memory core and the peripheral circuitry. The custom SRAM ICs allow an intermediate level of controllability although they are limited to one process node and are hard to implement.

Each of the proposed experiments and their results are discussed in detail in the subsequent chapters.

Table 1.1: Comparison of testing platforms.

Platform	**Control**	**Scalablity**	**Match to Reality**
SPICE Simulation	Best	Least	Least
Commercial SRAM IC	Least	Intermediate	Best
Custom SRAM IC	Intermediate	Best	Best

1.6 Thesis Outline

The thesis is organized into six chapters. Chapter two reviews the SRAM architecture and explains the working of the SRAM Power-up PUF. Chapter three shows the experiment setup and results for the SPICE simulations of SRAM bit cells and SRAM models. Chapter four explores the effect of supply ramp-up time on commercially available SRAM Integrated Circuits, and compares to the results from Chapter three. Chapter five focuses on the implementation of a custom SRAM IC. It explains the different stages of the design implementation in detail and the testing methodology used to test the chips that were taped out. Finally, chapter six concludes the thesis and discusses future work.

CHAPTER 2

SRAM PUFS

This chapter explains the architecture of an SRAM module and its operations. The chapter also explains the use of SRAM cells as Physically Unclonable Functions. We also discuss the proposed hypothesis and compare the platforms we use to test our hypothesis.

2.1 SRAM Architecture

An SRAM module consists of a memory core, row circuitry, column circuitry and control logic. Figure 2.1 portrays the overall architecture of the SRAM module. The memory core is made of SRAM cells packed as rows and columns. Each SRAM cell is usually made of six transistors, four NMOS and two PMOS. Four transistors (M0-M3) are arranged as cross coupled inverters and two more (M4-M5) that work as pass transistors. Since a cross-coupled inverter has two stable states, each cell stores 1 bit of information. When the wordline WL is not asserted, the cross-coupled inverters maintain the value as long as power is supplied to them.

The row circuitry consists of row decoders and wordline drivers. The row decoders select the wordline that must be activated based on the address provided and the wordline drivers activate the wordline based on the control circuitry. The row circuitry is pitch-matched to the SRAM cells to ensure that each wordline has the same drive strength across the array.

The column circuitry consists of pre-charge circuitry, column decoders, write drivers and sense amplifiers. The pre-charge circuitry consists of PMOS transistors

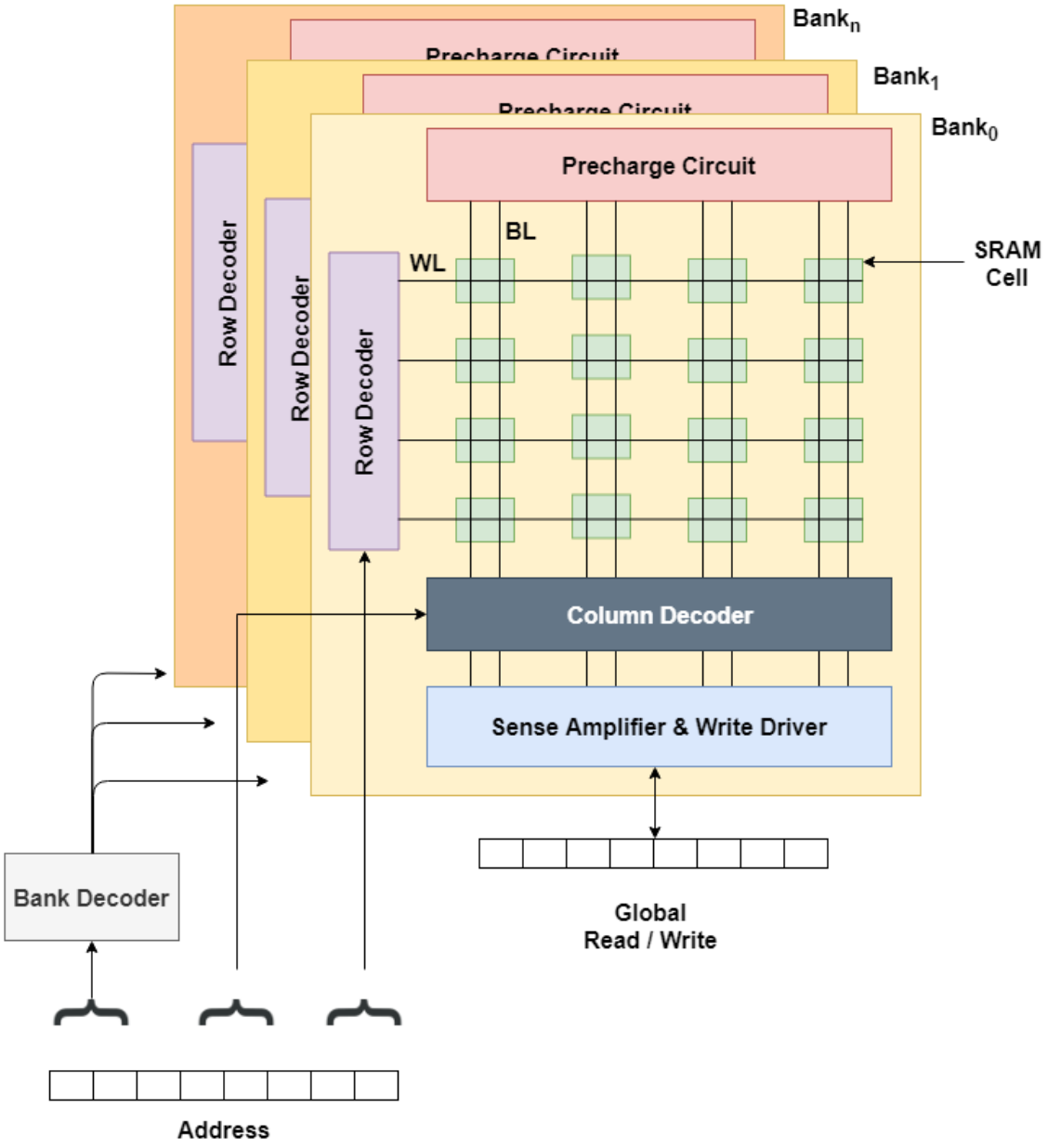


Figure 2.1: Architecture of a SRAM module.

to charge the bitlines BL and $\overline{BL}$ before a read and write operation. The column decoders are similar to the row decoders and are used to access particular bits in a row of the memory core. The write drivers input values to a particular bit cell by driving bitlines BL and $\overline{BL}$. The sense amplifiers are placed after the write drivers

at the end of the bitlines, and are used to read out values by sensing the differential voltages on the bitlines. Figure 2.2 shows a single column consisting of the pre-charge circuit and the word driver.

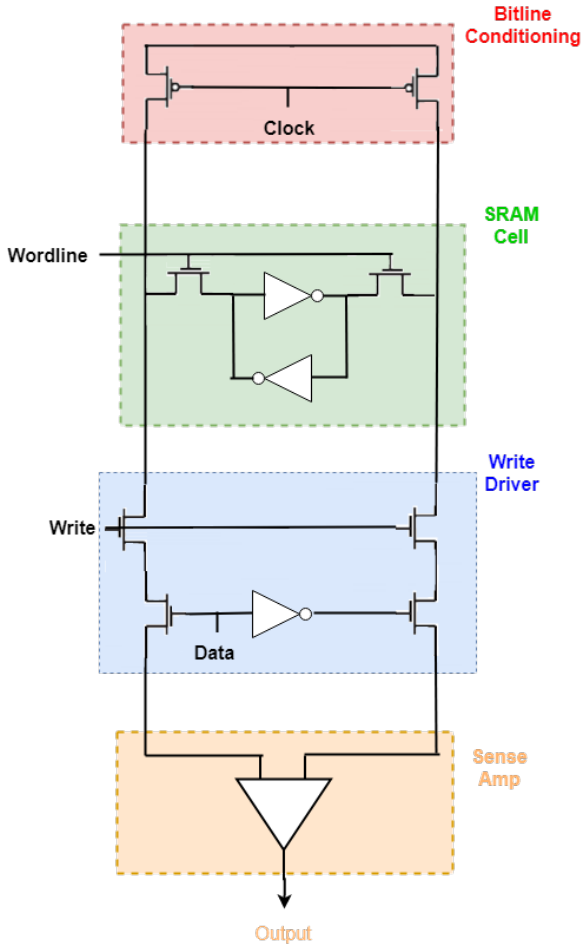


Figure 2.2: A slice of a SRAM column showing the bitline conditioning and write drivers [54].

The read operation is achieved by first pre-charging the bitlines. The pre-charge circuitry conditions the bitlines to ensure both BL and $\overline{BL}$ are at the same voltage. Once both the bitlines are at the same potential, the row decoder and wordline driver activate the wordline corresponding to the address selected. Depending on the value stored in the cell, BL and $\overline{BL}$ are driven to opposite polarities. This small change in

potential between the bitlines are sensed and amplified by the regenerative feedback of sense amplifiers.

The write operation is conducted by first precharging the bitlines and then selectively pulling one of them low. The write driver is used to determine and pull either BL or $\overline{BL}$ low depending on the value to be written to the cell. The row circuitry then activates the wordline that corresponds to the address where the value should be written. The regenerative feedback of the cross-coupled inverters forces the bit value written on the bitlines into the cell.

2.2 Operation of Power-Up SRAM PUF

The idea behind the power-up SRAM PUF is that values held in the SRAM cells on power-up are unique to each IC and are repeatable over time for a particular PUF. As the SRAM is volatile, the values stored are lost after power is removed. Thus, when the SRAM power is removed and reapplied, each bit cell (shown in Figure 2.3) in the SRAM powers up into a 0 or a 1. If the probability of the SRAM cell to power-up to a 0 is p_0, then the probability to power-up to a 1 is $p_1 = 1 - p_0$. Even if the inverters forming each SRAM bit cell is intended to be of equal strength and drive, process variations during manufacturing results in one of the inverters being stronger than the other. This causes the particular bit cell to have a higher affinity to settle into one state over the other, i.e., $p_0 > p_1$ or $p_0 < p_1$. If sufficient number of cells' power-up values are measured, the collection of the power-up affinities form a unique fingerprint to the SRAM module.

As the power-up state is a function of both process variation and noise present in the circuit during power-up, the power-up values measured can be different. Based on the influence of noise on the power-up state of the cell and its affinity to power-up into a 0 or a 1 bit, SRAM cells can be categorized into strong-0 cell, strong-1 cell or a matched-cell. If the process variation mismatch is large between the inverters of the

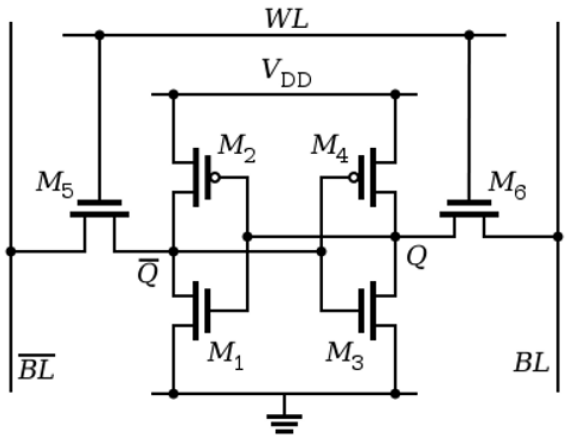
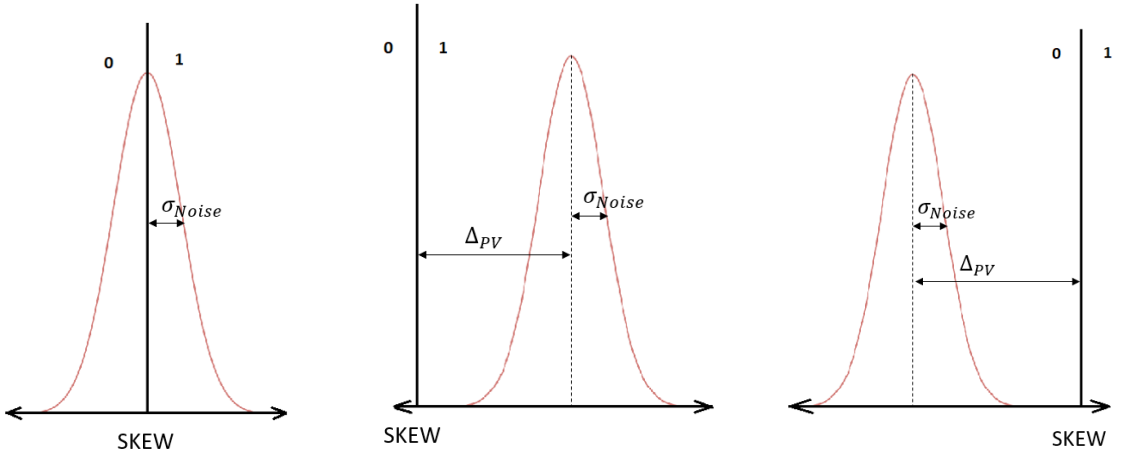


Figure 2.3: A single SRAM cell [54].

cell, the effect of noise on the power-up is minimized and power-up value is dependent only on the process variation. For a given cell, if $p_0 \approx 1$, the cell is categorized as a strong-0 cell and if $p_1 \approx 1$, the cell is categorized as a strong-1 cell. On the other hand, matched-cells have an equal tendency to power-up into 0 or 1 ,i.e., $p_0 = p_1 = 0.5$. Figure 2.4 shows the three types of bit cells based on power-up values.



(a) Matched Cell (b) Strong-1 Cell (c) Strong-0 Cell

Figure 2.4: Figure showing effect of process variation and noise on the power-up state. A matched cell's power-up state is influenced by the noise while a strong-1 and strong-0 cell's power-up is independent of noise.

Since the power-up states of strongly skewed cells are independent of noise and are dependent only on the process variation, these power-up states can be used as identifying fingerprints and to generate reliable secret keys [23].

2.3 Reliability Metric

As a PUF uses a physical measurement to generate keys, noise in the underlying physical processes lead to errors in key reconstructions. To ensure that SRAM PUFs can be used for key generation, it is necessary for the SRAM PUF responses to be repeatable over different conditions. In the case of power-up SRAM PUF, the secret key generated using the power-up states of a given PUF must be reliable over multiple power-ups. Reliability measurements determine the amount of repeatability of the PUF responses over multiple trials. The reliability is measured by calculating the Hamming distance between the PUF responses.

First, for a given power-up j the N-bit SRAM PUF response, denoted as R_j, is obtained. To overcome the errors in responses, multiple PUF responses $\{R_0, R_1, ..., R_m\}$ are collected and averaged over an odd number of trials m to produce the enrollment power-up response R. Each bit of the enrollment response $R[i]$ is calculated using equation 2.1.

$$R[i] = \begin{cases} 0, & \text{if } \frac{1}{m}\sum_{j=0}^{m} R_j[i] < 0.5 \\ 1, & \text{otherwise} \end{cases} \tag{2.1}$$

When the same PUF's response are observed under a new trial p, the difference between the enrollment response R and the new response R_p gives the Bit Error. The total number of bit positions that differ between the two responses is called the Hamming Distance and is calculated using equation 2.2.

$$HD(R, R_p) = \sum_{i=0}^{N-1} R[i] \oplus R_p[i] \tag{2.2}$$

The Within Class Hamming Distance (WCHD) is the average Hamming Distance obtained over k trials for the same PUF instance and is given by equation 2.3.

$$WCHD(R) = \frac{1}{k} \sum_{j=0}^{k-1} HD(R, R_j) \tag{2.3}$$

For the same PUF, the ideal reliability is 100%, meaning that the Hamming Distance between the golden fingerprint R and a new fingerprint R_p is 0.

2.4 Uniqueness Metric

Although we want the PUF responses to be reproducible over multiple trials, i.e., be reliable, we want the PUF responses from different devices to be significantly different from each other. The PUF response must ideally be completely random between devices and this randomness is measured by calculating the Hamming Distance between PUF response of two different PUFs. If the power-up response of two PUFs A and B are R_a and R_b, the Hamming distance between the responses is calculated by equation 2.4.

$$HD(R_a, R_b) = \sum_{i=0}^{N-1} R_a[i] \oplus R_b[i] \tag{2.4}$$

where $R[i]$ represents the i^{th} bit of the response R. The Between Class Hamming Distance (BCHD) is calculated using equation 2.5.

$$BCHD(a, b) = \frac{1}{\frac{m*(m-1)}{2}} \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} HD(R_{a,i}, R_{b,j}) \tag{2.5}$$

where the subscript a and b represent two different PUFs A and B, i represents the i^{th} trial and j represents the j^{th} trial. The Between Class Hamming Distance (BCHD) is calculated over m trials.

Ideally, the BCHD is 50% meaning that half of the bits of the response of PUF A is expected to be different from that of PUF B.

CHAPTER 3

SPICE SIMULATIONS

In this chapter, we discuss two different SPICE simulation setups and the results obtained from the simulations. The objective behind performing SPICE simulation is to get an understanding of the effect of supply ramp on the SRAM PUF at a circuit level. To achieve this we first analyze the effect of supply ramp time on the SRAM bit cell under ideal conditions by grounding the wordlines and bitlines. As SRAM circuitry contains several components apart from the bit cells, we incorporate the peripheral circuitry in the next set of experiments to analyze the effect of supply ramp time in a more realistic manner. The SPICE simulations are performed using commercial 16nm FinFET technology models that incorporate parameters to reflect the inter-die and intra-die process variations found on the Silicon chips. While the SRAM bit cell models are provided by a foundry, the circuit level netlist of the SRAM module is provided by ARM. These SRAM modules contain a memory core of size 256x16 and include precharge circuitry, sense amplifiers, row decoders, wordline drivers and word drivers.

3.1 SRAM Bit Cell Simulations

To test the effect of supply ramp-up time on the power-up state of SRAM cells, we create a SPICE netlist containing a 6T SRAM cell as shown in Figure 2.3. The transistor sizing in the netlist is provided by the foundry and the wordline WL, bitlines BL and $\overline{BL}$ are grounded. To introduce process variation among different instances of SRAM cells, we use Monte Carlo simulation to induce process variation into the

SRAM cells. The process variation in the devices are modeled by expressing key model features as equations of five principal parameters. Since the models are proprietary, we do not have explicit control over the parameters nor do we have information about what each of the parameters control. For each Monte Carlo simulation, each of the five variables are sampled for each transistor, thereby creating a unique SRAM cell. This is shown in Figure 3.1.

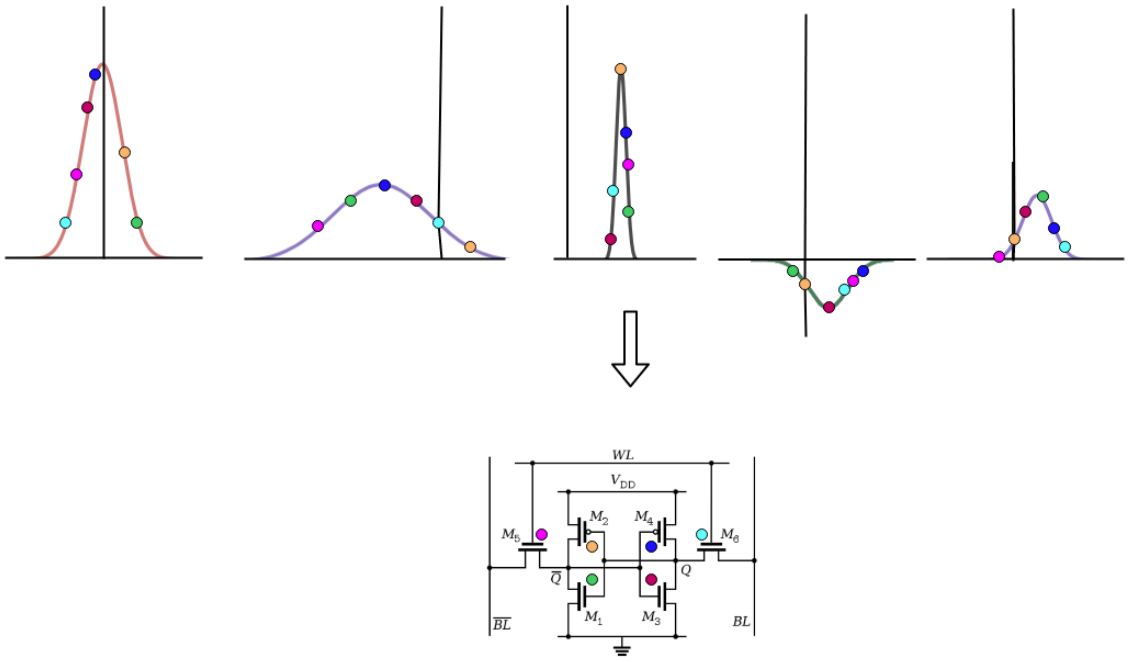


Figure 3.1: Five parameters following a Gaussian distribution are sampled for each transistor in every Monte Carlo simulation. The distributions shown are for illustration purposes and the actual Gaussian distribution of each parameter is unknown to us.

The power supply to each of the cells is provided with a particular ramp-up time and the final power-up state of each cell is recorded. Apart from incorporating process variation through Monte Carlo simulation, we add transient thermal noise to the nodes of the SRAM cell as shown in Figure 3.2. The thermal noise is modelled using a Gaussian distribution with a variance of σ^2_{Noise} [22], where temperature is T and the capacitance at the node is C (see Equation 3.1).

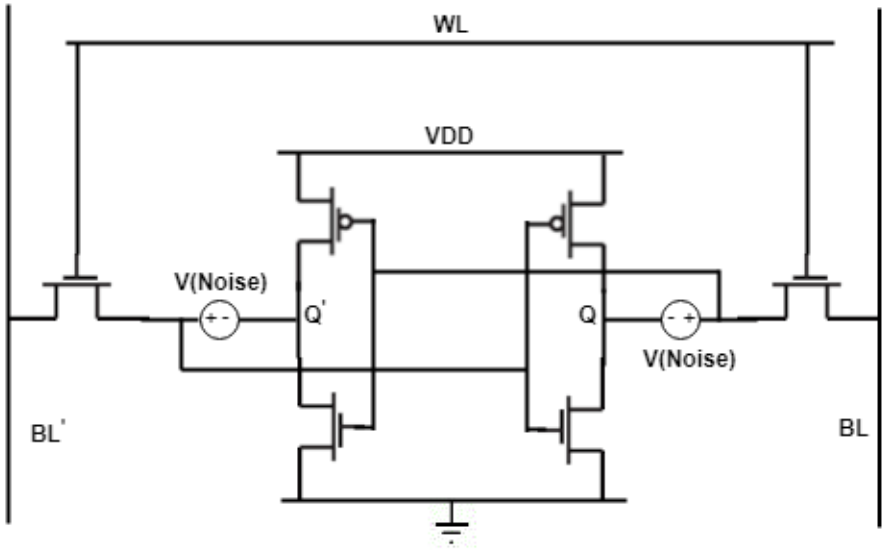


Figure 3.2: Schematic of the SRAM bit cell with noise added.

$$\sigma^2_{Noise} = \frac{2K_BT}{C} \tag{3.1}$$

The distribution is sampled at each time step of the simulation and added to the nodes through a voltage source. Figure 3.3 shows power-up traces of the SRAM cells with noise introduced to the nodes. We observe that different SRAM cells stabilize at different voltage based on the process variation present in the cell. From the waveform we see that the supply voltage plays a role in determining the power-up state of the cell. Thus we experiment with different supply ramp times to see its effect on the SRAM.

The Bit Error Rate is calculated by finding the difference between the power-up states of the cells in two simulations with different transient thermal noise added to the nodes. Figure 3.4 shows the concept utilized to find the BER. As measurements can get biased due to the nature of random noise we add to the SRAM cells, we randomly choose SRAM cells from individual trials to find the BER. The algorithm

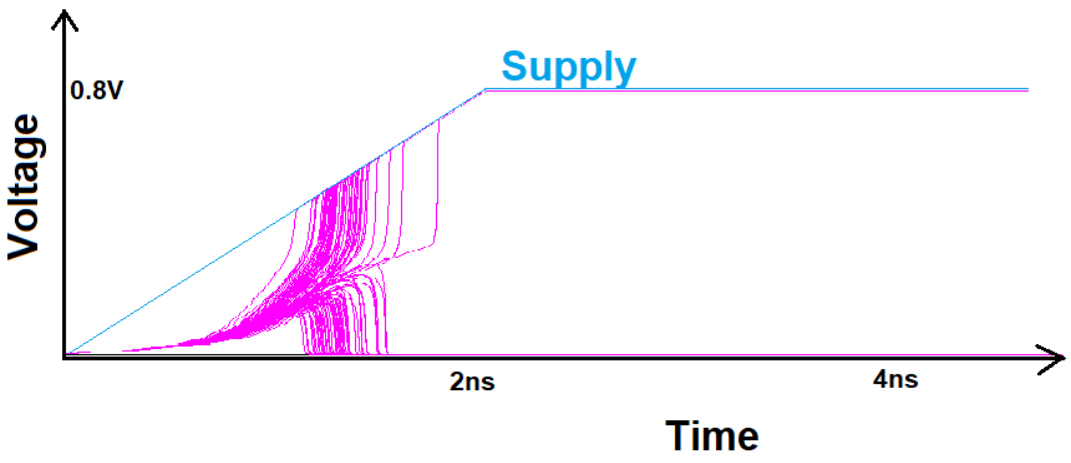


Figure 3.3: Plot depicting the power-up states of SRAM cells using 100 Monte Carlo simulations with transient noise added.

we use to find the BER across multiple trials at each ramp time of the power supply is shown in Algorithm 1 .

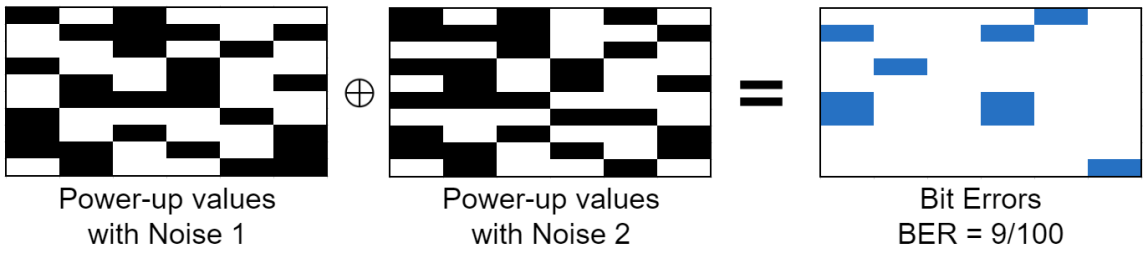


Figure 3.4: The difference between the power-up states with different noise gives the BER.

We measure the power-up states of SRAM cells using 1000 Monte Carlo simulations over 10 trials. In each trial we use a unique seed for generating the Gaussian noise although the distribution has the same mean and deviation. Thus each trial provides 1000 single-bit power-up values and we get 10,000 power-up values in total per ramp time. The BER measurements for 1000 Monte Carlo Simulations are shown

Algorithm 1 Pseudo code for calculating reliability of SRAM bit cells

$N \leftarrow$ Number of Power-Up Trials (N=10)
$M \leftarrow$ Number of bit cell instances from Monte Carlo simulations (N=1000)

for `i in N` **do**
    $Noise_i \leftarrow random(seed = i)$
    for `j in M` **do**
        $Trial_i[j] \leftarrow Power - Up[Noise_i]$

Bit Errors = 0
for $1, 2, \cdots,$ `count` **do**
    $a \leftarrow random(0, N)$
    $b \leftarrow random(0, N) \neq a$
    $index \leftarrow random(0, M)$
    Bit Errors += $Trial_a[index] \oplus Trial_b[index]$
BER = Bit Errors/(count)

in Figures 3.6a, 3.6b and 3.6c with a linear ramp in the power supply. We see that the BER reduces marginally ($\sim 1\%$) when the ramp times are increased.

Apart from the linear ramp waveform, a sigmoid supply signal was also used to test the effect of ramp time on the reliability of the SRAM cells. A sigmoid function was chosen to measure the effect of a non-linear supply signal and also to use a signal that does not have inflection points. The sigmoid function used in our experiments is defined in Equation 3.2 using the error function described in Equation 3.3.

$$V(t) = 0.4V + 0.4 * erf(\alpha t - 5) \tag{3.2}$$

$$erf(x) = \frac{1}{\sqrt{\pi}} \int_{-x}^{x} e^{-t^2} dt \tag{3.3}$$

We vary the value of α to change the slope and the time that the sigmoid signal takes to reach VDD. Figure 3.5 shows the sigmoid signal for $\alpha = 1$ and $\alpha = 10$. The value of the sigmoid signal at each time step was calculated using a Python script and these values were provided as a piece-wise linear signal in the SPICE netlist.

The BER measured between trials of noisy simulations at the same ramp time using Algorithm 1 is shown in Figures 3.6d, 3.6e and 3.6f.

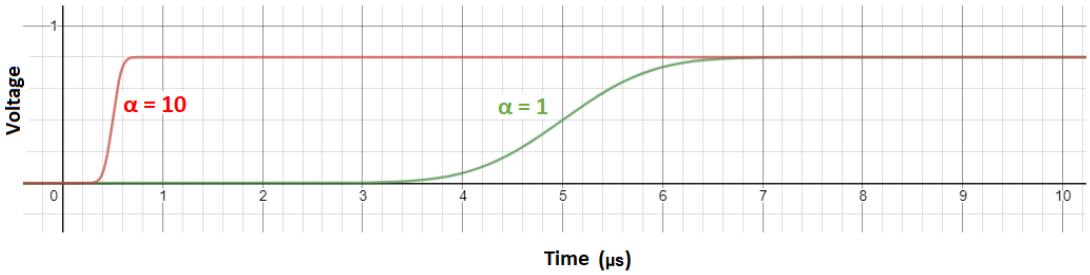
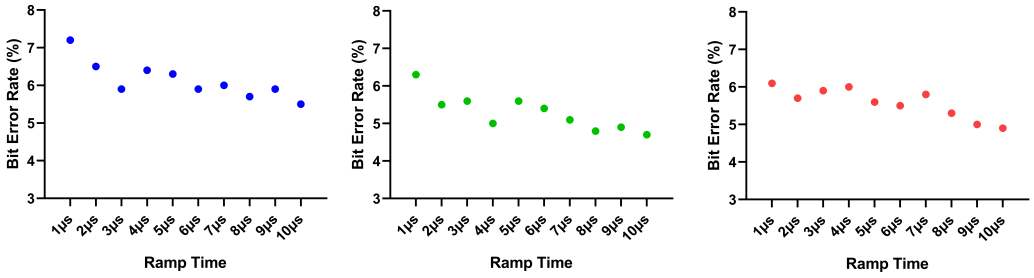
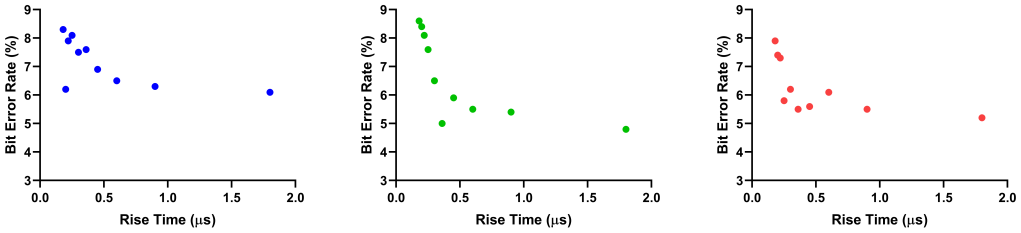


Figure 3.5: The sigmoid function used as the supply signal for the SPICE simulation.



(a) 10°C (b) 25°C (c) 40°C



(d) 10°C (e) 25°C (f) 40°C

Figure 3.6: BER of SRAM bit cells at different ramp times and temperatures. Figures 3.6a, 3.6b, 3.6c use linear power-up supply while Figures 3.6d, 3.6e, 3.6c use sigmoid power-up.

3.2 SRAM Module Simulation

SPICE simulations of SRAM cells assumes ideal conditions and thus may not be representative of the behaviour of a SRAM module. While we assume that the bitlines and wordlines are grounded during power-up in our previous experiment, in reality the peripheral circuitry of the SRAM may impact the voltage of the bitlines and the wordlines during the power-up process. Thus, we analyze the effect of supply ramp times on the SRAM cells with the peripheral circuitry included. We use the SPICE model of the SRAM module produced by the ARM Memory Compiler. The SPICE model contains a memory core of 256x16 size, sense amplifiers, write drivers, precharge circuitry and wordline drivers along with control circuitry. Although additional measurements and experiments would help conclusively understand the effect of the peripheral circuitry on the power-up values, we show preliminary results obtained in this section. The main limitation of our experiments is caused by the large number of transistors in the SRAM module. Due to this, we add noises to a limited set of nodes in the circuit, as well as increase the simulation time step to generate results.

3.2.1 BER between noise-free and noisy power-up

The first experiment we perform is to measure the difference in power-up between the noise-free SRAM module and that of the circuit with transient thermal noise added to the nodes of the circuit. We perform this measurement with the power supplies to the peripheral circuitry and the memory core tied together and again with the power supplies separated. The separated supplies allows the power-up of the SRAM module to be sequenced. We first power-up the peripheral circuitry and power up the core supply once the peripheral supply is stable as shown in Figure 3.7.

We perform noise-free SPICE simulation on the SRAM module with the supplies tied together and record the power-up values of all cells in the module across different

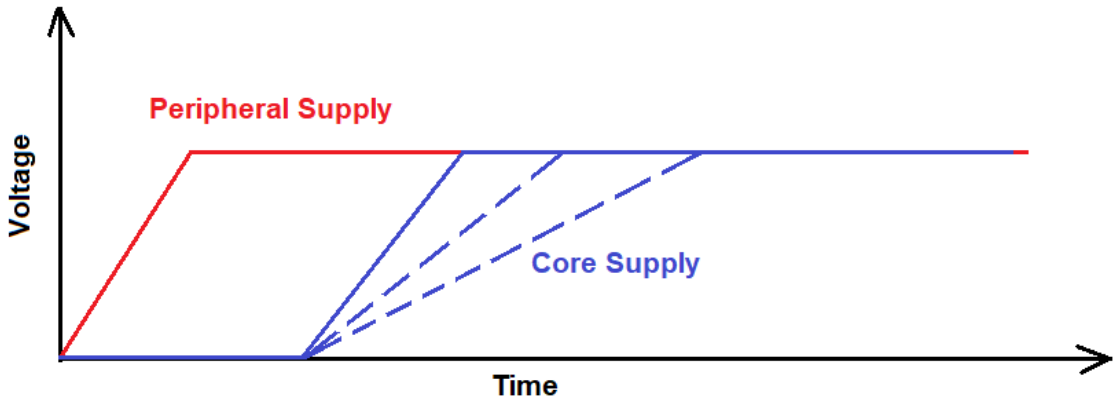
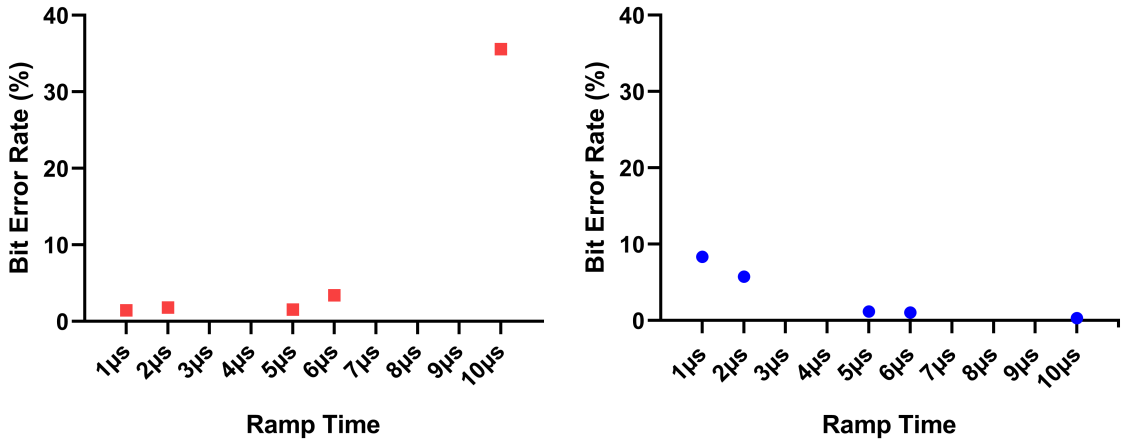


Figure 3.7: The separation of power supplies to the peripheral circuitry and the memory core allows the power-up to be sequenced. First the peripheral circuitry is powered up and then the memory supply is ramped up.

ramp times. 10 Monte Carlo simulations were used to introduce process variation to the transistors of the SRAM module, thus providing ten instances of 256x16 modules, with 25x16x10 single-bit power-up values. We next repeat the simulation with noise added to the nodes of the circuit. To reduce the run time of the simulation we add noise only to the SRAM core, sense amplifiers, wordlines and bitlines. The difference between the power-up values of the noise-free and noisy trial provides the BER[1]. Figure 3.8a shows the BER measured when the power supplies of the SRAM are tied together.

We repeat the above BER experiment but instead with the power supplies separated and sequenced. The peripheral circuitry is always ramped up to VDD in the first 1 µs. We then ramp up the memory core at varying ramp times to perform the BER measurements. Figure 3.8b shows the BER when the power supplies are sequenced. Due to the limited number of measurements, limited noise sources and

[1]BER is typically measured between trials of noisy instances. Thus, BER calculated between the noise-free and noisy trial is an approximation. The simulation with noise consumes significant run time as compared to noise-free trials.



(a) Tied power supplies.

(b) Sequenced power supplies.

Figure 3.8: BER for the SRAM module measured between noise-free and noisy power-up values.

increased simulation step size, the results obtained may not necessarily capture the behavior we would observe in taped-out ICs or other simulations. We observe that on average, the BER measurements for the tied power supplies is higher than that of the sequenced power supplies.

3.2.2 BER across ramp times for noise-free power-up

Another experiment we perform is to find the change in the noise-free power-up of the SRAM module when the power supplies are sequenced as compared to the tied supplies. This experiment allows us to analyze the effect of separating the power supplies. We achieve this by finding the difference in power-up values between different ramp times. The algorithm used for these measurements is shown in Algorithm 2. Figure 3.9a shows the errors calculated between two trials at different ramp times when the power supplies at tied together. Figure 3.9b shows the same measurement when the power supplies are sequenced. We notice that the difference between the

power-ups from different ramp times shows that separating and sequencing the power supplies may cause power-up values to be less sensitive to supply ramp changes.

Algorithm 2 Algorithm for calculating the change in noise-free power-up

$N \leftarrow$ Number of module instances through Monte Carlo simulation (N=10)
$M \leftarrow$ Number of Addresses in the module (M=256)
Ramps $\leftarrow$ List of Ramp Times

for `R in Ramps` **do**
    for `i in N` **do**
        for `j in M` **do**
            $Trial_{R,i}[j] \leftarrow Data[j]$ collected after ramp time of R

for $R_1 \neq R_2 \subset$ Ramps **do**
    $\text{Errors}_{R_1,R_2} = 0$
    for $1, 2, \cdots,$ `count` **do**
        $a \leftarrow random(0, N)$
        $Address \leftarrow random(0, M)$
        $\text{Errors}_{R_1,R_2} \mathrel{+}= \sum_{bit=0}^{15} Trial_{R_1,a}[Address][bit] \oplus Trial_{R_2,a}[Address][bit]$
    $\text{BER}_{R_1,R_2} = \text{Errors}_{R_1,R_2}/(\text{count} * 16)$

3.3 Conclusion

Results of the SPICE simulation of the SRAM cells imply that there is an improvement in the reliability of SRAM cells' power-up values when the ramp-up time of the supply is increased (see Figure 3.6). We also observe that the trend is similar even when the supply is non-linearly ramped up. Thus, increasing the rise time of the power supply to the core could make the SRAM bit cells more reliable in powering-up to the same value.

From the SPICE simulations of the SRAM modules, we observe that separating and sequencing the power supplies to the peripheral circuitry and memory core increases the reliability of the SRAM module with increasing ramp times (see Figure 3.8). Separating and sequencing the power supplies also ensures a smaller change in the power-up values of the SRAM (see Figure 3.9). Although more measurements

Ramp	1µs	2µs	5µs	6µs	10µs
1µs		9.33	8.92	8.96	9.66
2µs			9.37	9.58	9.52
5µs				9.73	8.82
6µs					9.92
10µs					

(a) Tied power supplies

Ramp	1µs	2µs	5µs	6µs	10µs
1µs		9.02	6.3	7.89	9.5
2µs			8.97	7.67	5.62
5µs				8.2	6.45
6µs					6.17
10µs					

(b) Sequenced power supplies

Figure 3.9: Percent change in power-up values across different ramp times.

are required to conclusively understand the effect of the peripheral circuitry on the power-up values, the preliminary experiments show that separating and sequencing the power supplies between the peripheral circuitry and the memory core may lead to an overall improvement in reliability of the SRAM PUF and reduce the change in power-up values between ramp times.

CHAPTER 4

COMMERCIAL SRAM IC TESTING

This chapter presents the experimental data from power-up states of commercially available SRAM ICs. We explore the effect of supply ramp-up time on these commercial SRAM ICs and compare it to the SPICE simulation results obtained for SRAM cells and modules. Testing commercial SRAM ICs helps us validate the effect of supply ramp-up when the power supplies of the memory core and peripheral circuit are tied together. Table 4.1 shows the commercial ICs that are used for performing the experiments.

Table 4.1: List of commercial SRAM ICs used in the experiment.

IC Name	**Manufacturer**	**Memory Size**	**Interface**
23LC1024 [50]	Microchip Technologies	128K x 8	Serial
AS6C6264 [33]	Alliance Memory	8K x 8	Parallel

To collect the power-up states of the commercial SRAM ICs, we use an "Arduino Mega 2560" [4] development board to drive the inputs and store the outputs. The 23LC1024 SRAM ICs require serial inputs and produce serial outputs. The Serial Peripheral Interface (SPI) module on the Arduino board is used to interface with the SRAM and read the data at each address sequentially. The AS6C6264 SRAM require parallel inputs and produce parallel outputs which are accomplished using Digital IOs of the Arduino development board. The outputs produced by the SRAM ICs are read by the Arduino board and stored into a file, which is later post-processed to find the reliability and uniqueness for each ramp-up time. Figure 4.1 shows the test setup used to readout power-up states from commercial SRAM ICs.

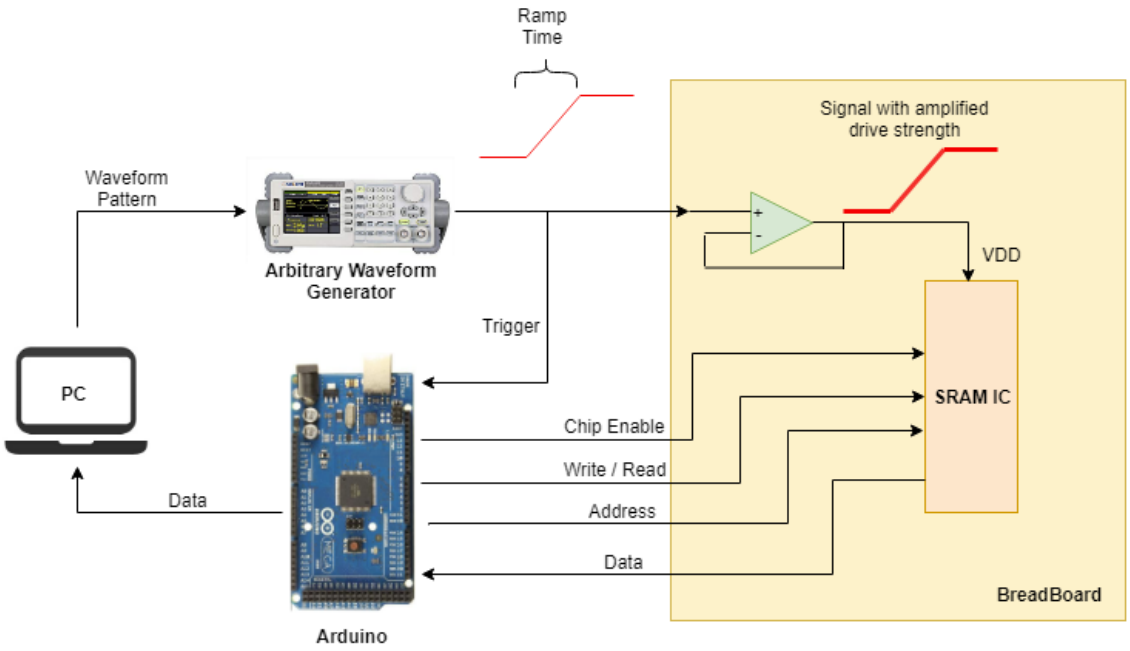


Figure 4.1: The setup for testing commercial SRAM ICs. The function generator provides different ramp-up times and the Arduino provides the inputs to the IC and stores the power-up states.

The Keysight 33511B arbitrary waveform generator [48] is programmed by using the Keysight Waveform Builder Pro software [49] to produce a waveform that ramps the supply voltage for a given time and stays at VDD until the Arduino collects all the power-up values from the SRAM IC. The Arduino board detects when the supply waveform has completed the ramp-up and enables communication with the SRAM once the supply signal is stable. As the signal generated by the waveform generator has a low drive strength, an operational amplifier (Op-Amp) [26] is used to increase the drive strength. The Op-Amp is configured as a voltage follower circuit and is powered by an external power supply.

4.1 Reliability Measurements

The power-up values at each address for the chip are collected as part of a trial. Thus, one trial for the 23LC1024 IC forms an array of 128K locations, each containing 8 bits while a trial for AS6C6264 forms an array of 8K locations with 8 bits each. The power-up values are collected N times for each ramp. These trials are used to calculate the BER at a given ramp using algorithm 3. In our experiments, each address contains 8-bits of information and $count = 10,000$, thus the Bit Error is averaged over 10,000 x 8 bits. The BER calculated for different ramp times is shown in Figure 4.2. The results show that BER on average is around 4% for the AS6C6264 SRAM and is around 5% for the 23LC1024 SRAM. Notably, the BER drops by 1% for the AS6C6264 SRAM when the supply ramp time is 1 second whereas the BER for the 23LC1024 SRAM is not impacted much with ramp times.

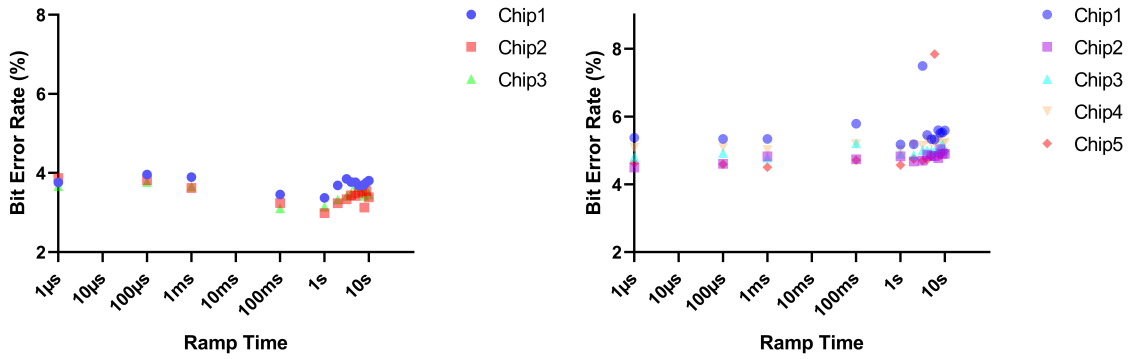
Algorithm 3 Pseudocode for calculating reliability of a single chip at a single ramp time

$N \leftarrow$ Number of Trials
$M \leftarrow$ Total Number of Addresses in IC

for `i in N` **do**
    for `j in M` **do**
        $Trial_i[j] \leftarrow Data[j]$

Bit Errors $= 0$
for $1, 2, \cdots,$ `count` **do**
    $a \leftarrow random(0, N)$
    $b \leftarrow random(0, N) \neq a$
    $Address \leftarrow random(0, M)$
    Bit Errors $+= \sum_{bit=0}^{7} Trial_a[Address][bit] \oplus Trial_b[Address][bit]$
BER = Bit Errors/(count $*$ 8)

Another analysis performed on the power-up values collected is to measure the reliability when the trials are conducted at different ramp times for the same chip. The measurement is achieved by using Algorithm 4. The cross-ramp time BERs are



(a) AS6C6264 SRAM BER measurement (b) 23LC1024 SRAM BER measurement

Figure 4.2: BER measurements of commercial SRAM ICs at various ramp times.

shown in Figure 4.3 and Figure 4.4. The experiments show that larger differences between the supply ramp time at the enrollment and the key regeneration phase leads to lower reliability. For example, for the AS6C6264 SRAM, the BER can be as high as 20% which is 5 times higher than the BER obtained for any fixed ramp time (see Figure 4.2a). For the 23LC1024 SRAM, the cross ramp BER can be more than 2 times that of the single ramp BER (see Figure 4.2b). Thus, ensuring that the supply ramp times are constant at the time of enrollment and key regeneration is important in minimizing BER.

Ramp Time	1µ	100µ	1ms	100ms	1s	2s	3s	4s	5s	6s	7s	8s	9s	10s
1µs		4.04	7.27	15.23	15.72	13.77	12.78	12.15	12.10	12.80	12.65	13.31	12.19	12.28
100µs	4.04		6.18	14.12	14.53	12.50	11.64	11.15	10.89	11.93	11.80	12.84	11.51	11.34
1ms	7.27	6.18		9.25	10.04	8.58	8.24	8.37	8.65	11.32	11.85	14.07	10.55	10.53
100ms	15.23	14.12	9.25		4.75	6.44	8.50	10.20	11.70	16.85	18.10	20.97	16.02	15.99
1s	15.72	14.53	10.04	4.75		4.76	6.88	8.93	10.44	15.74	17.29	20.35	15.16	15.19
2s	13.77	12.50	8.58	6.44	4.76		4.23	5.60	7.04	12.21	13.49	16.52	11.29	11.39
3s	12.78	11.64	8.24	8.50	6.88	4.23		4.00	4.86	9.50	10.77	13.76	8.67	8.62
4s	12.15	11.15	8.37	10.20	8.93	5.60	4.00		3.94	7.67	8.67	11.53	6.76	6.75
5s	12.10	10.89	8.65	11.70	10.44	7.04	4.86	3.94		6.39	7.16	10.01	5.48	5.48
6s	12.80	11.93	11.32	16.85	15.74	12.21	9.50	7.67	6.39		4.38	5.93	4.34	4.42
7s	12.65	11.80	11.85	18.10	17.29	13.49	10.77	8.67	7.16	4.38		4.95	4.05	4.04
8s	13.31	12.84	14.07	20.97	20.35	16.52	13.76	11.53	10.01	5.93	4.95		6.20	6.10
9s	12.19	11.51	10.55	16.02	15.16	11.29	8.67	6.76	5.48	4.34	4.05	6.20		3.69
10s	12.28	11.34	10.53	15.99	15.19	11.39	8.62	6.75	5.48	4.42	4.04	6.10	3.69	

Figure 4.3: The average BER for AS6C6264 SRAM chips for different enrollment and key regeneration supply voltage ramp times.

Algorithm 4 Pseudocode for calculating reliability at different ramp times

$N \leftarrow$ Number of Trials
$M \leftarrow$ Total Number of Addresses in IC
Ramps $\leftarrow$ List of Ramp Times

for `R in Ramps` **do**
    for `i in N` **do**
        for `j in M` **do**
            $Trial_{R,i}[j] \leftarrow Data[j]$ collected after ramp time of R

for $R_1 \neq R_2 \subset$ Ramps **do**
    $\text{Errors}_{R_1,R_2} = 0$
    for $1, 2, \cdots,$ `count` **do**
        $a \leftarrow random(0, N)$
        $b \leftarrow random(0, N)$
        $Address \leftarrow random(0, M)$
        $\text{Errors}_{R_1,R_2} \mathrel{+}= \sum_{bit=0}^{7} Trial_{R_1,a}[Address][bit] \oplus Trial_{R_2,b}[Address][bit]$
    $\text{BER}_{R_1,R_2} = \text{Errors}_{R_1,R_2}/(\text{count} * 8)$

Ramp Time	1µs	100µs	1ms	100ms	1s	2s	3s	4s	5s	6s	7s	8s	9s	10s
1µs		5.11	5.42	6.98	9.46	10.32	10.26	10.21	10.50	11.17	11.32	11.46	10.91	11.23
100µs	5.11		5.10	6.65	8.88	9.77	9.55	9.77	9.93	10.59	10.76	10.90	10.31	10.54
1ms	5.42	5.10		6.05	7.86	8.66	8.48	8.43	8.90	9.51	9.62	9.72	9.21	9.43
100ms	6.98	6.65	6.05		6.12	6.94	7.13	7.13	7.45	8.06	8.11	8.26	7.86	8.06
1s	9.46	8.88	7.86	6.12		5.16	5.66	5.99	6.09	6.61	6.39	6.54	6.52	6.51
2s	10.32	9.77	8.66	6.94	5.16		5.40	5.68	5.81	6.13	5.95	6.03	6.28	6.17
3s	10.26	9.55	8.48	7.13	5.66	5.40		5.34	5.40	5.79	5.66	5.70	5.96	6.04
4s	10.21	9.77	8.43	7.13	5.99	5.68	5.34		5.06	5.48	5.32	5.39	5.78	5.81
5s	10.50	9.93	8.90	7.45	6.09	5.81	5.40	5.06		5.41	5.24	5.33	5.65	5.73
6s	11.17	10.59	9.51	8.06	6.61	6.13	5.79	5.48	5.41		5.41	5.39	5.95	5.92
7s	11.32	10.76	9.62	8.11	6.39	5.95	5.66	5.32	5.24	5.41		5.15	5.56	5.49
8s	11.46	10.90	9.72	8.26	6.54	6.03	5.70	5.39	5.33	5.39	5.15		5.53	5.50
9s	10.91	10.31	9.21	7.86	6.52	6.28	5.96	5.78	5.65	5.95	5.56	5.53		5.11
10s	11.23	10.54	9.43	8.06	6.51	6.17	6.04	5.81	5.73	5.92	5.49	5.50	5.11	

Figure 4.4: The average BER for 23LC1024 SRAM chips for different enrollment and key regeneration supply voltage ramp times.

4.2 Uniqueness Measurements

Similar to reliability measurements, uniqueness measurements are performed on the data collected for each supply ramp time. But unlike reliability measurements where we choose trials from the same chip, we now choose trials from two different chips for a given ramp time. The procedure for calculating the uniqueness is given in Algorithm 5. Figure 4.5 shows the measured uniqueness averaged across different

chips at different ramp times. We observe that the uniqueness for the AS6C6264 SRAM reduces by $\sim 10\%$ when the ramp time is around 1 second (see Figure 4.5a). This minimum coincides with the BER minima found previously in Figure 4.2a. When increasing the ramp times above 1 second, the uniqueness of the SRAM increases to the ideal value of 50%. For the 23LC1024 SRAM, the minimal uniqueness coincides with the lowest BER found for the supply ramp time of 1 µs (see Figure 4.2b). Increasing supply ramp times increases the uniqueness up to 4%. The change in uniqueness is however much smaller in comparison to the AS6C6264 SRAM.

Algorithm 5 Pseudocode for calculating uniqueness at a given ramp time

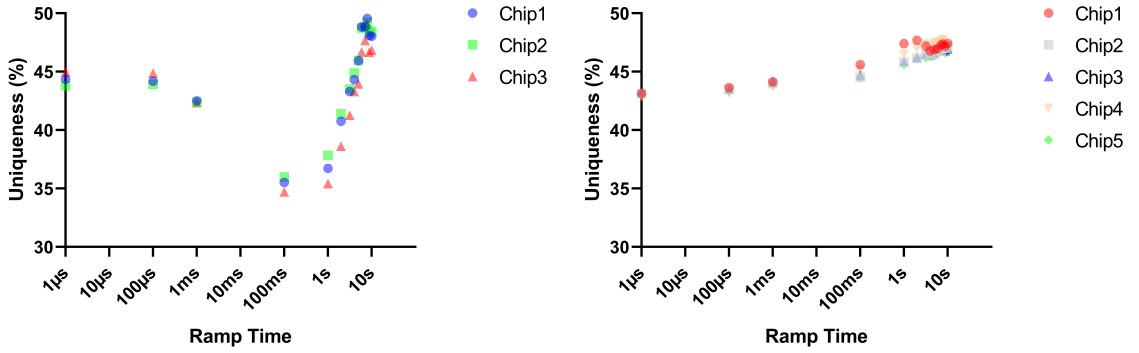
$N \leftarrow$ Number of Trials
$M \leftarrow$ Total Number of Addresses in IC
Chips $\leftarrow$ List of different chip instances

for `C in Chips` **do**
    for `i in N` **do**
        for `j in M` **do**
            $Trial_{C,i}[j] \leftarrow Data[j]$ collected from Chip C

for $C_1 \neq C_2 \subset$ Chips **do**
    $\text{Difference}_{C_1,C_2} = 0$
    for $1, 2, \cdots,$ `count` **do**
        $a \leftarrow random(0, N)$
        $b \leftarrow random(0, N)$
        $Address \leftarrow random(0, M)$
        $\text{Difference}_{C_1,C_2} \mathrel{+}= \sum_{bit=0}^{7} Trial_{C_1,a}[Address][bit] \oplus Trial_{C_2,b}[Address][bit]$
    $\text{Uniqueness}_{C_1,C_2} = \text{Difference}_{C_1,C_2}/(\text{count} * 8)$

4.3 Power-Up Bias of the IC

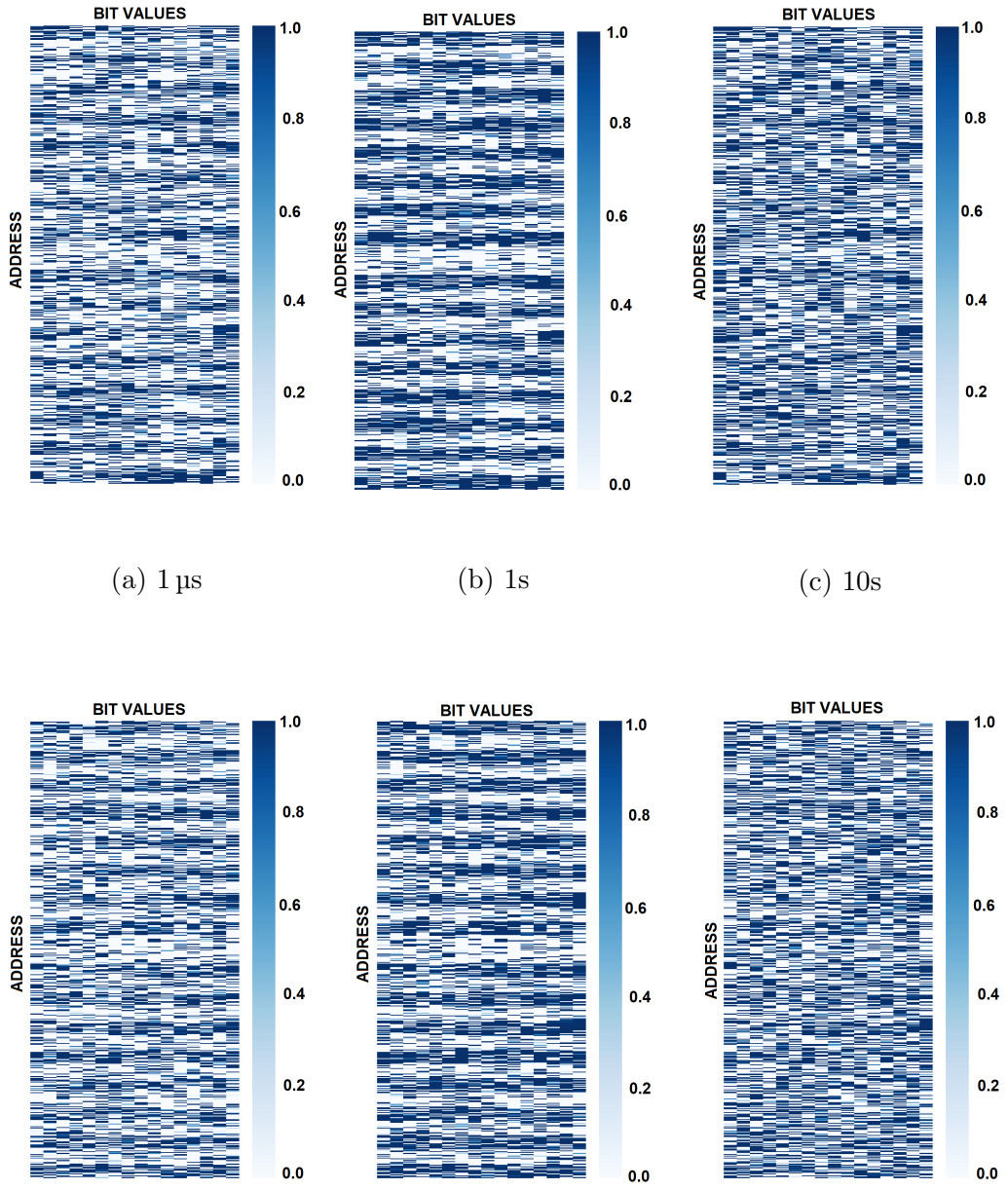
To observe the effect of supply ramp on the power-up values across the chips, we plot the average value of each bit in a subset of the address space for each ramp time. Figures 4.6a, 4.6b, 4.6c show the average power-up values on a particular AS6C6264 SRAM at three different ramp times. We observe that particular addresses of the



(a) (b)

Figure 4.5: Figure 4.5a shows the uniqueness of the AS6C6264 chips and Figure 4.5b shows the uniqueness of the 23LC1024 chips.

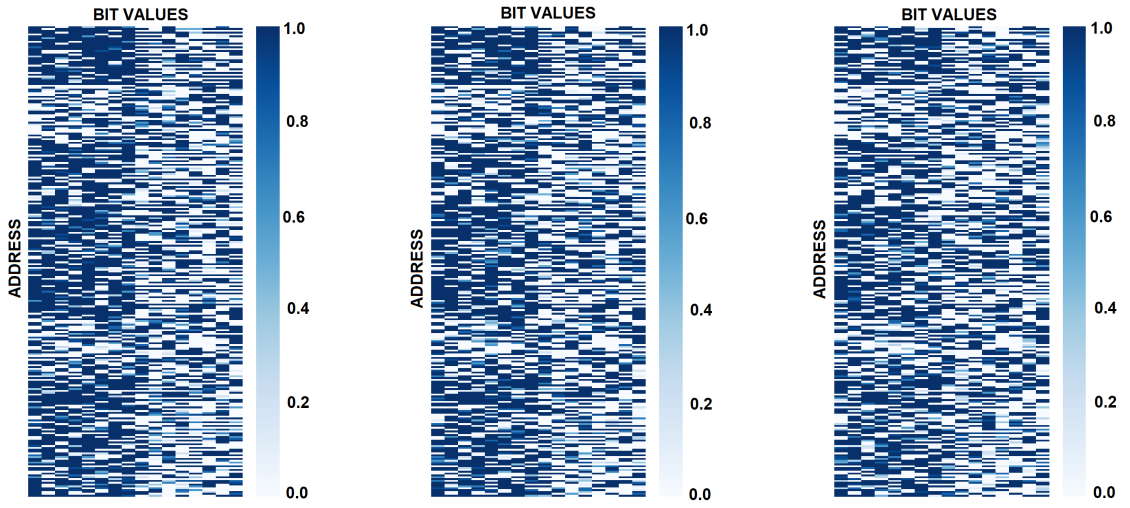
IC get biased to power-up in the 1-state over 0-state when the supply ramp time is increased while others get biased to 0-state. We term this as the "local-bias" where particular addresses have a high tendency to start-up in the 1-state. Figures 4.6d, 4.6e and 4.6f show the local bias on another instance of the AS6C6264 IC. Due to the repeatability of the effect across chip instances and across address space, this is likely caused by the architecture of the IC. We see a similar local bias in the 23LC1024 IC, as shown in Figure 4.7. For the AS6C6264 SRAM, this local bias is most prominent around 1 seconds while the local bias is strongest around 1 µs for the 23LC1024 SRAM. This is the reason we see a reduction in the uniqueness of the chips at those ramp time. Beyond this ramp time, the local bias effect across the address space is reduced although the overall percentage of bits starting in 1-state continues to increase. Figure 4.8 shows the overall percentage of bits in the SRAM powering-up in the 1-state across different supply ramp times.



(a) 1 µs (b) 1s (c) 10s

(d) 1 µs (e) 1s (f) 10s

Figure 4.6: Figures 4.6a, 4.6b and 4.6c show the average power-up values for one instance of AS6C6264 SRAM chips at three different ramp times, while Figures 4.6d, 4.6e and 4.6f belong to another instance of the same model. Each point's (X,Y) coordinates indicate its address and bit position in the address.



(a) 1 µs (b) 1s (c) 10s

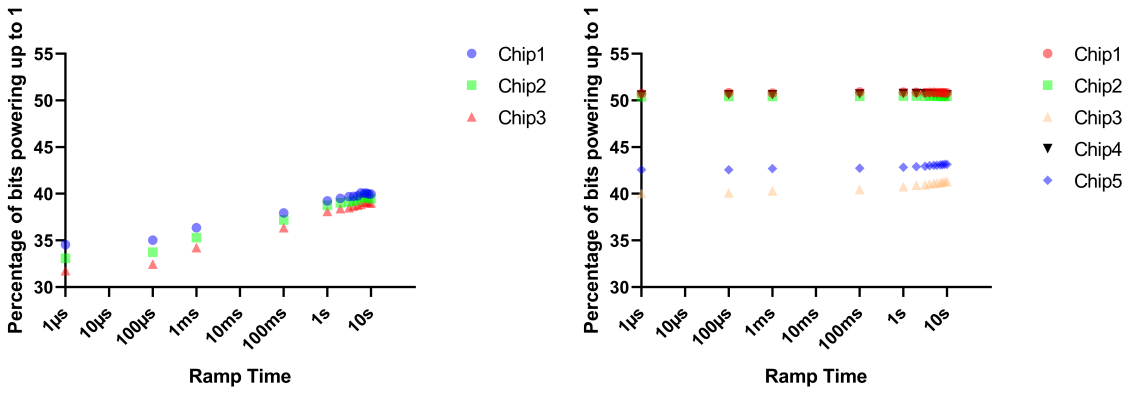
Figure 4.7: Figure showing the average power-up values for one of the "23LC1024" SRAM chips at three different supply ramp times.

4.4 Conclusions

In this chapter, we observe the following effects of the supply ramp time on the reliability, uniqueness and bias of the SRAM:

- Depending on the architecture of the SRAM, varying supply ramp time causes certain addresses to get biased to power-up in the 1-state (local bias) and others to 0-state (Figure 4.6 and 4.7). The ramp time at which this local bias is strongest most likely depends not just on the architecture but also the technology node of the SRAM.

- When the local bias across the SRAM is maximum, the same set of addresses have a high probability of powering up to 1 across chip instances. This leads to a reduction in uniqueness (see Figure 4.5a and Figure 4.5b).



(a) AS6C6264

(b) 23LC1024

Figure 4.8: The bias measurement shows of the percentage of bits powering-up to 1-state for the two SRAM models.

- The ramp times that induce the strongest local bias also maximize reliability (see Figure 4.2a and Figure 4.2b).

From these observation we can conclude the following:

- Depending on the SRAM, we can choose a supply ramp time to either increase uniqueness or to increase reliability. For example, if we want to ensure a reliable SRAM PUF by using the AS6C6264 SRAM, we should choose a ramp time of 1 second (see Figure 4.2a) while maximizing uniqueness would require a ramp time of 10 seconds (see Figure 4.5a).

- Given that the reliability does not worsen significantly at larger ramp times, it is more beneficial to choose a supply ramp time to increase uniqueness than to increase reliability. For example, the difference in maximum and minimum BER is $\sim$ 2% while the difference in maximum and minimum uniqueness is $\sim$ 15% for the AS6C6264 SRAM (see Figures 4.2a and 4.5a).

- It is critical for reliability to use a consistent ramp time for enrollment and regeneration. If the enrollment was performed at a particular ramp time and the

regeneration is performed at a different ramp time, the BER increases rapidly with difference in ramp times (see Figures 4.3 and 4.4).

CHAPTER 5

CUSTOM SRAM IC

In the previous chapters we have seen that increasing ramp-up times on the power supply improves the reliability of the SRAM PUF. We also observe that separating the power supplies to the memory core and the peripheral circuitry further improves the reliability. Commercial SRAM ICs often have the power supplies to the peripheral circuitry and memory core tied together. Thus, to test our hypothesis in Silicon, we designed and taped-out a custom IC with separated supplies in a commercial 16nm FinFET technology. Although the taped-out chip was found to have a short on a peripheral circuitry power pin, the design steps taken for taping out lay a firm foundation for future work. The various aspects of designing the custom SRAM IC are described in the following sections.

5.1 Design and Synthesis

The custom SRAM IC design comprises four SRAM modules. Each SRAM module consists of 256 addresses each storing 16 bits of information. The SRAM modules are generated through the ARM Artisan Memory Compiler [25]. The compiler directives are configured to generate independent power pins for the SRAM memory core and SRAM peripheral circuitry. The SRAM module contains pins as shown in Table 5.1. Due to a constraint on the number of available IO pins, we design the primary address and data inputs and the primary data output to be 1-bit wide. The serial input signals are converted internally to provide the parallel inputs required by the SRAM modules. Similarly the parallel outputs generated by the SRAM modules are

converted back to a serial output stream as shown in Figure 5.1. Multiplexer and demultiplexers are used to test one SRAM module at a time. The controller is a state machine which triggers on the positive edge of the external clock input and helps interface with the SRAM modules.

Table 5.1: Pin functionality and widths of the ARM Compiler generated SRAM module.

PIN Name	Functionality	Width
A	Address	8
D	Data In	16
Q	Data Out	16
CLK	Clock	1
CEN	Chip Enable	1
GWEN	Write or Read	1
VSSE	Ground	1
VDDPE	Peripheral Power	1
VDDCE	Core Power	1

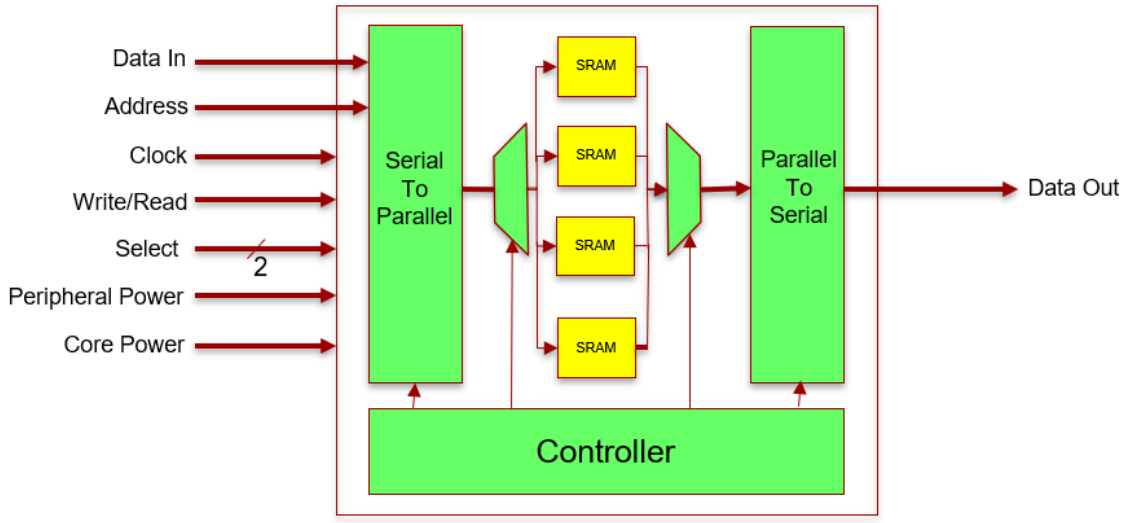


Figure 5.1: Block diagram of the custom SRAM. Components in yellow are generated by the ARM Artisan Memory Compiler and those in green are synthesized.

In Figure 5.1, the control circuitry (shown in green) is designed using Verilog and the SRAM modules (shown in yellow) are generated using the ARM Memory Compiler. The functional correctness of the design is verified by performing logic simulation using Mentor ModelSim. Using Verilog models for the SRAM modules generated by the ARM Memory Compiler, we use a testbench to write data into random locations of the SRAM and verify correctness by reading back the data.

Synopsys Design Compiler is used to perform logic synthesis and generate a gate level netlist of the design. ARM Standard cell library along with the Verilog RTL code is used to create the gate level netlist. The standard cell library contains timing, area and power information of the standard cells characterized at various input slews and output loads. The synthesis tool uses this information to optimize the design subject to constraints. Along with the netlist, Synopsys DC generates timing, area and power reports. These reports can be used to ensure that the design meets all the constraints imposed on it. Further, these reports can be used as initial constraints during Placement and Routing. Initial timing checks on setup and hold violations are performed at this stage along with clock skew and available timing slack calculation. Figure 5.2 shows the design flow to synthesize the netlist from the Verilog RTL code. The reports from the synthesis flow are shown in Table 5.2.

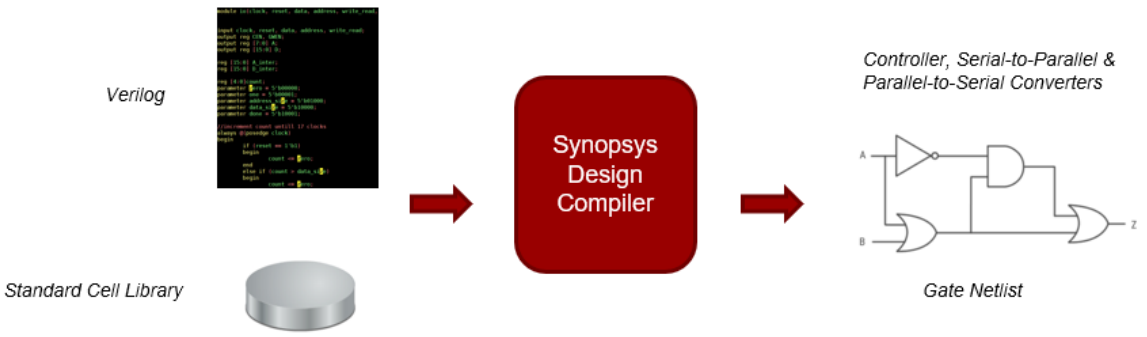


Figure 5.2: The design flow for generating the gate level netlist from the Verilog design.

Table 5.2: Synthesis Reports generated from Synopsys DC.

Parameter	Reported Values
Combinational Cell Count	698
Sequential Cell Count	420
Utilization	40.26%
Cell Rows	455
Cell Internal Power	376.5 µW
Net Switching Power	135.4 µW
Total Dynamic Power	511.9 µW
Cell Leakage Power	111.5 µW
IO Pad Power Consumption	62%
Register Power Consumption	27%
Combination Logic Power Consumption	1%
Slack (for 5000ns period)	2224.62 ns
Critical path length	108.19 ns

5.2 Floorplanning

The first step in the physical design flow is to create a floorplan using Synopsys IC Compiler. The floorplan comprises of IO area and core area. The IO area is allocated for the placement of IO pads while the core area contains the standard cells, hard macros, signal and power routing used in the design. The size of the core area is controlled by setting a target utilization and an aspect ratio. Utilization is the ratio of used area to that of available area. IO pads are placed outside the core boundary at a distance called the core to IO clearance. These pads are used to connect the primary inputs and outputs from the core area to the external package pins via bumps.

Based on the size of the IO pad ring and the core boundary, design floorplans are either pad limited or core limited. The floorplan of a core limited design is determined by the size of the core boundary. These designs have high core utilization but lower utilization in the IO pad ring. However, pad limited designs have low core utilization and the IO pad ring determines the overall floorplan size. Our design is pad limited

due to the large number of IO pads used relative to the core area. Figure 5.3 shows the floorplan depicting the core area, the IO area and the SRAM macros.

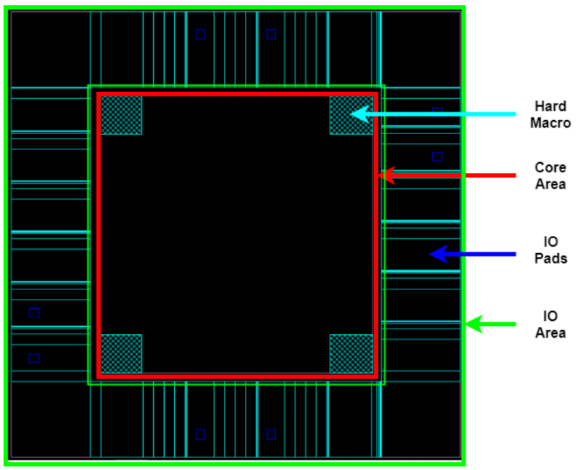


Figure 5.3: The floorplan showing the core boundary and the IO pads.

The next step in the physical design flow is power grid creation. The power grid routes power and ground signals from the IO pads to the various parts of the design. In a typical SRAM IC a single power domain is used to deliver power to all cells and macros as depicted in Figure 5.4a. The power network consists of two dimensional grids where the top grid is routed in the two highest metals to minimize IR drop. The lower grids is a fine grained mesh and is intended to meet the $\frac{di}{dt}$ requirements. Each of the grids contain its own power straps which are spaced evenly across the network. The power ring is used to ensure symmetric power delivery to all regions of the design from the external IO power pad.

In our custom SRAM design, to supply power to the SRAM core, we create one power network VDDCE around each SRAM module as illustrated in Figure 5.4b. This network has a power ring routed using Metal 9 in the vertical direction and Metal 8 in the horizontal direction. The top grid of the network uses Metal 8 and Metal 9 while the lower grid uses Metal 5 and Metal 4. The standard cells and the peripheral circuitry of all SRAM modules share a single power network VDDPE, whose power

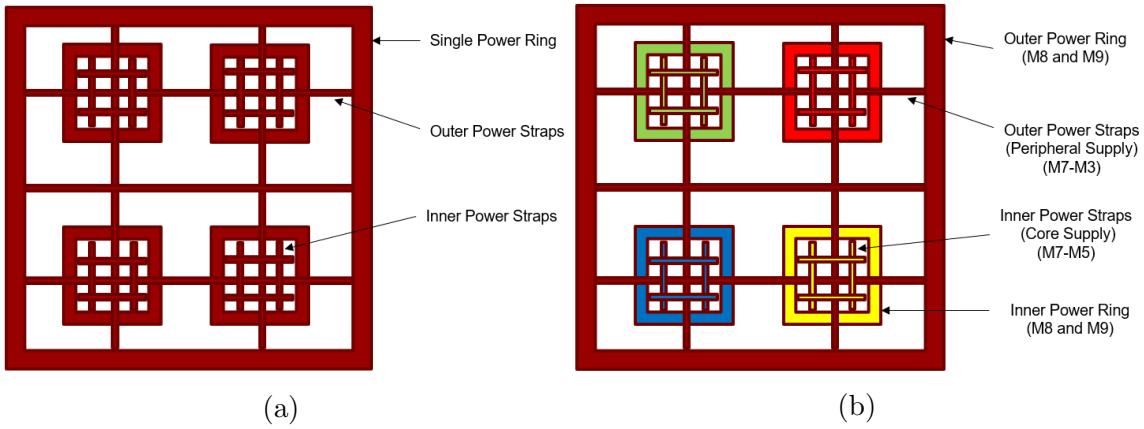


Figure 5.4: Figure 5.4a shows the typical power grid using a common power ring to the entire IC. Figure 5.4b shows the power grid of the custom SRAM IC using one power ring around the entire design and one around each SRAM module.

ring and grids use metals similar to that of the VDDCE network. One single power network VSS is used to provide ground to the entire design and is similar in structure to the VDDPE network. Figure 5.5 shows all the power networks around a single SRAM module.

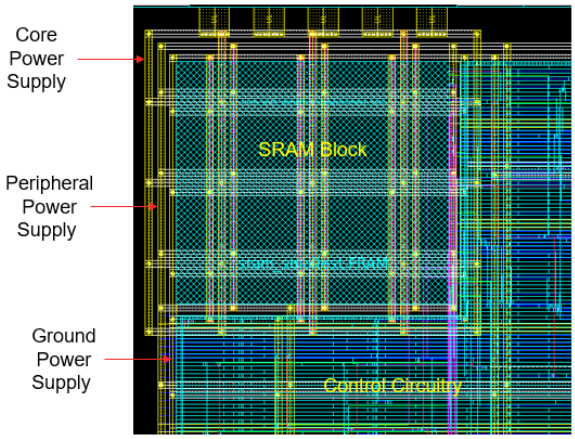


Figure 5.5: Floorplan of the design depicting power rings and straps forming the grid.

5.3 Placement, CTS and Routing

The next stage in the design process after the floorplanning is to determine the position of standard cells and hard macros. This process is called placement. The standard cells are placed in rows of fixed heights in the core area. The rows are mirrored horizontally to share power and ground straps between adjacent rows. The SRAM modules are hard macros which are placed at the four corners of the floorplan. This results in the FRAM view of the design. A FRAM view is an abstract model of the layout information. It does not contain all the layer information, but only contains pin positions, metal blockages and via blockages of the standard cells and macros.

After placement, Clock Tree Synthesis (CTS) is performed to pre-route the clock signal from the external IO pin to the pins of the standard cells and SRAM macros. Figure 5.6 shows the clock tree of the design. After CTS, global and detailed signal routing is performed using a grid based router. Figure 5.7 shows the overall design flow during placement, CTS and routing. Following each of the above stages, ICC performs area and timing optimization.

After the design is placed and routed, the FRAM views of the standard cells and the SRAM macros are replaced with layer information. The design flow at this stage is depicted in Figure 5.9. The standard cell layer information is contained in ARM standard cell libraries. The SRAM macro layout is generated by the ARM Memory Compiler and is used at this design stage. The SRAM macro layout is shown in Figure 5.8a. The SRAM macro shows the organization of the various parts of the SRAM used in the design. The core consists of four banks made of 6T bit cells. Each of the 16-bit words in the core is addressed through a row decoder that converts a 8-bit address input to activate one of 256 wordlines. The sense amplifiers read out the value of the bits stored in the banks onto the data-out signal.

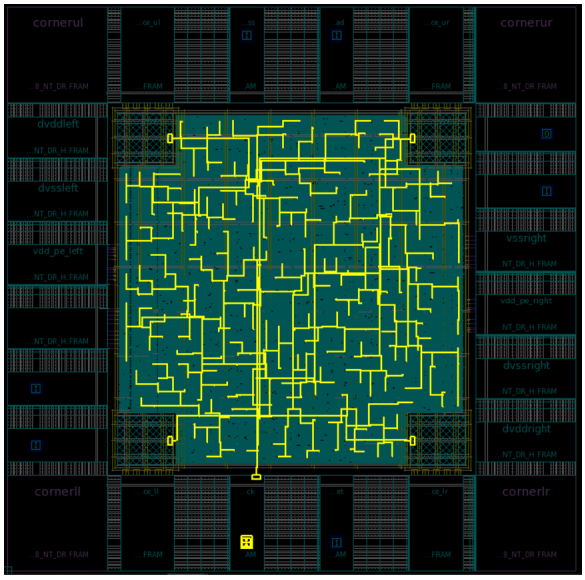


Figure 5.6: The design with the Clock Tree highlighted.

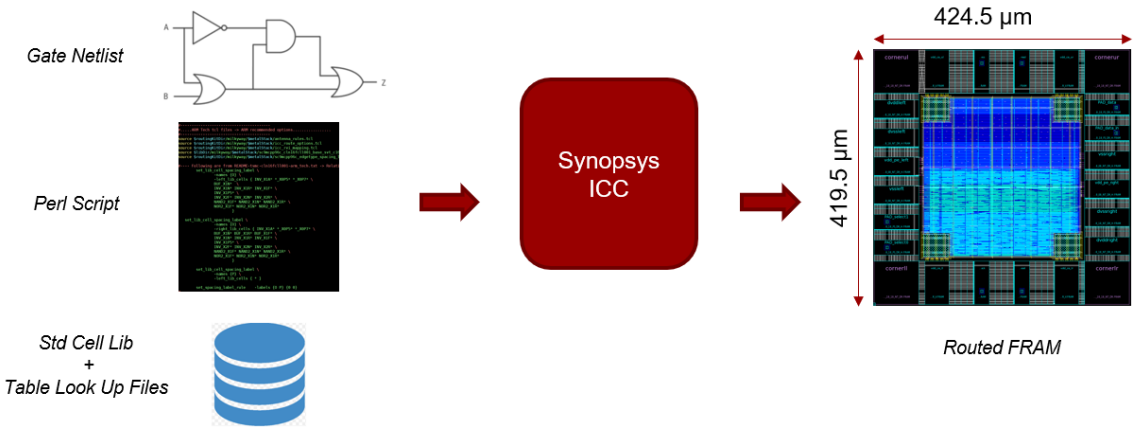
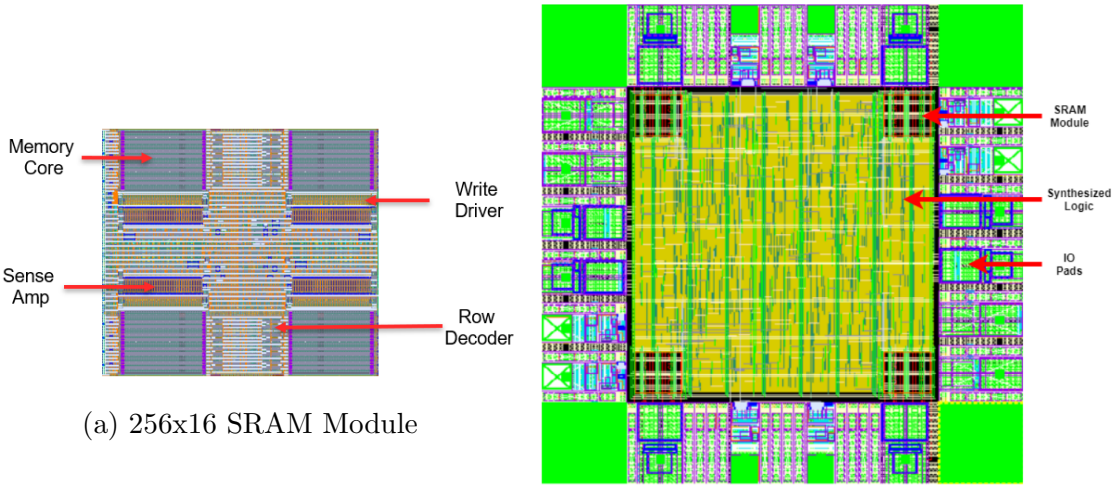


Figure 5.7: The design flow for generating a placed and routed FRAM view of the design.

The layout of the custom SRAM design which consists of the control circuitry and four SRAM modules is shown in Figure 5.8b. The table 5.3 shows the dimensions of the SRAM module, the design and the entire chip. After the FRAM views of the standard cells and macros are replaced with the layer information to form the GDS2 layout, Design Rule check (DRC), Layout Vs Schematic (LVS) and Electrical Rule



(a) 256x16 SRAM Module

(b) Design Layout

Figure 5.8: Figure 5.8a shows the GDS2 layout of a single SRAM module. Figure 5.8b shows the GDS2 layout of the custom SRAM design.

Check (ERC) are performed on the design. The error-free design was integrated onto the layout that was sent to the fabrication plant.

Table 5.3: Dimensions of the various modules in the design.

Module	**Dimensions** (µm x µm)
SRAM Module	36 x 40
Design	420 x 425
Chip	2500 x 2500

5.4 Chip Integration

Two instances of the custom SRAM design are integrated into the taped-out chip to allow for intra-die testing and reliability comparisons as shown in Figure 5.11a. Three other designs were implemented by my colleagues on the same chip as well. The IO pads of the SRAM designs are routed to the bumps in the Re-Distribution Layer (RDL). Figure 5.10 shows the RDL bumps and the integration to the final

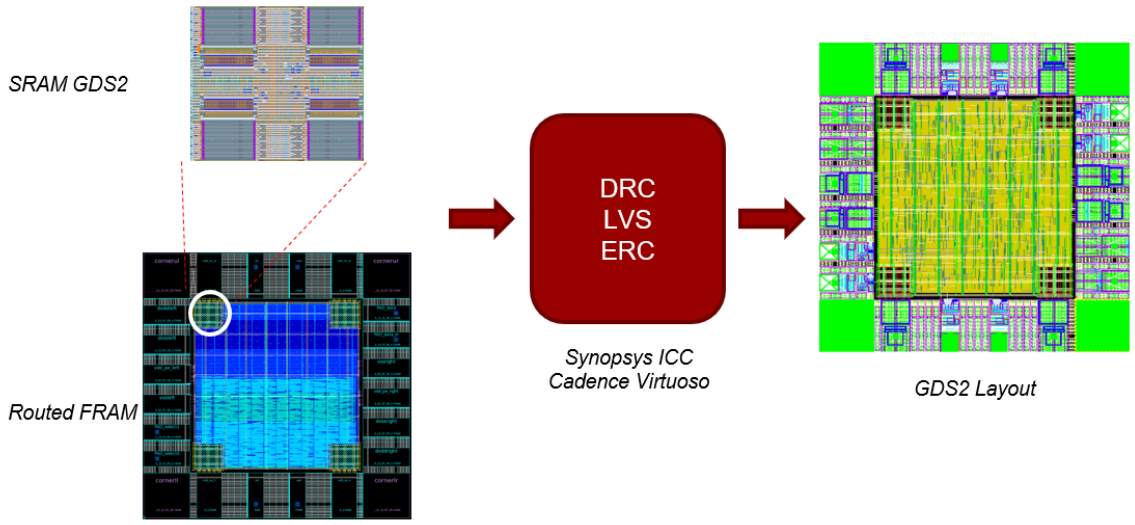


Figure 5.9: The design flow for generating the design layout.

packaging. The RDL routing was also performed using Synopsys ICC. With all the designs integrated with the RDL routing, the GDS2 layouts were tested for DRC, LVS and ERC using Mentor Calibre. Figure 5.11a shows the two instances of the design along with the RDL routing. The taped-out chip dimensions are 2.5mm x 2.5mm. Figure 5.11b shows one of the SRAM designs on the fabricated die.

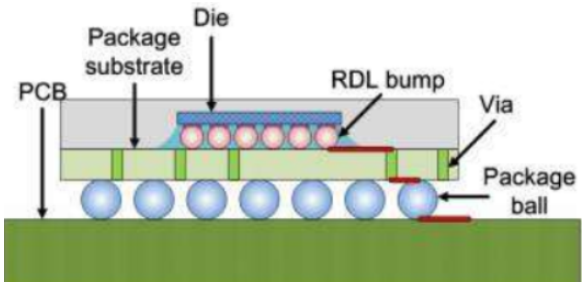
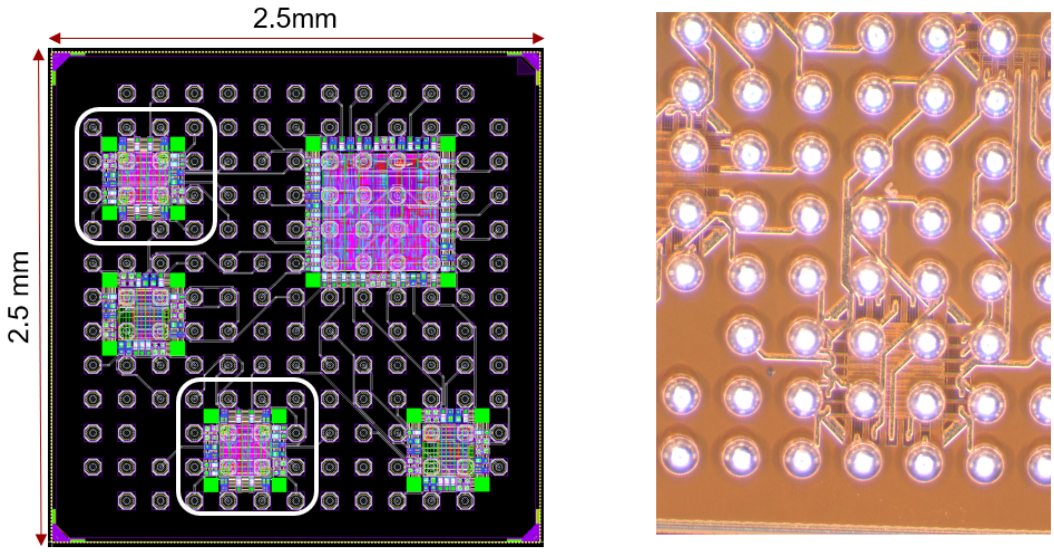


Figure 5.10: Figure showing how the die IO pads are connected to the package balls [51].

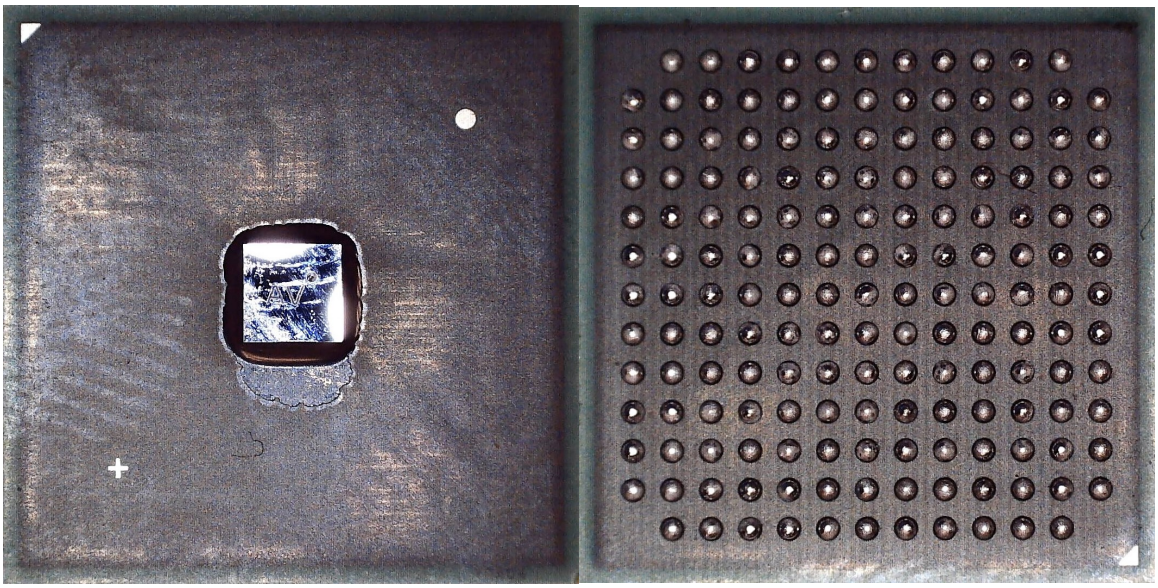


(a) (b)

Figure 5.11: Figure 5.11a displaying the layout with all the designs integrated. The SRAM designs are highlighted using a white box. The regularly placed white circles are the RDL bumps. Figure 5.11b shows a photograph of a part of the unpackaged die showing the custom SRAM design, RDL bumps and the RDL routing.

5.5 Chip packaging and PCB Design

The 2.5mm x 2.5mm die from the foundry is packaged as a Flip Chip Ball Grid Array (FC-BGA). The solder balls of the packaged chip have a 1mm pitch and are arranged into a 13 x 13 grid. Figure 5.12 shows the packaged chip and the balls of the chip. To connect the chip to signal and power pins, a Printed Circuit Board (PCB) with Surface Mount Technology (SMT) pads is required. The custom PCB is designed using Autodesk Eagle and uses 4 layers to route the balls of the chip to the pin header.



(a) Front side of the packaged chip (b) Back side of the packaged chip

Figure 5.12: The front side shows the die containing the designs while the back side shows the balls of the package arranged into a 13 x 13 grid.

To minimize the number of PCBs required to test the packaged chips, a BGA socket is incorporated into the PCB. The socket is made of a fixed bottom half which is mounted onto the PCB using drill holes and bolts. The exploded view of the socket is shown in Figure 5.13. The chip's BGA comes into contact with the PCB through an elastomer guide made of Gold wires embedded in an insulating plastic. The corresponding part on the PCB where the bottom half of the socket mounts

has SMT pads on the top layer to provide connectivity between the PCB and the elastomer guide. The top of the socket is a removable lid that is operated using a torque driver.

The 169 SMT pads corresponding to the 169 BGA are routed to 100 pin headers to create a breakout board. To achieve the routing from inner SMT pads, dogbone vias are used to route pads to the inner layers of the PCB. Figure 5.14a shows the SMT pads and the dogbone vias. Figure 5.14b shows the entire PCB with the pin headers and the routing from the BGA socket to the pin headers.

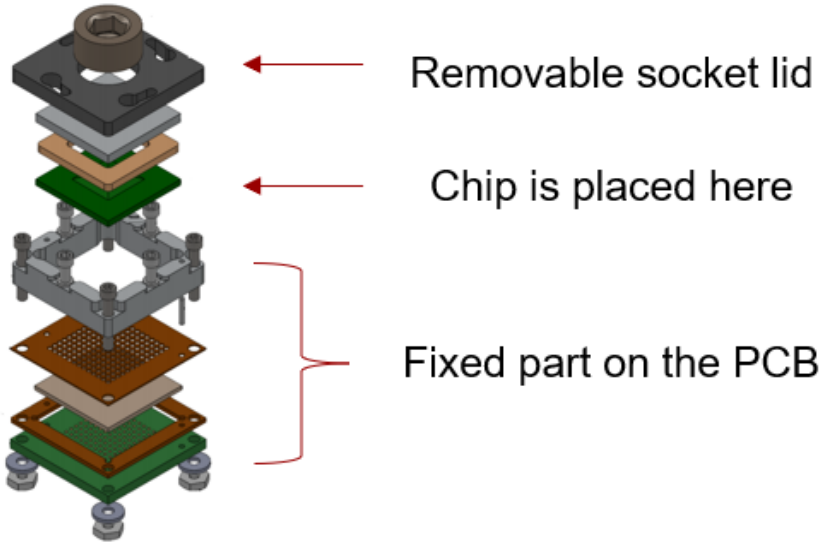
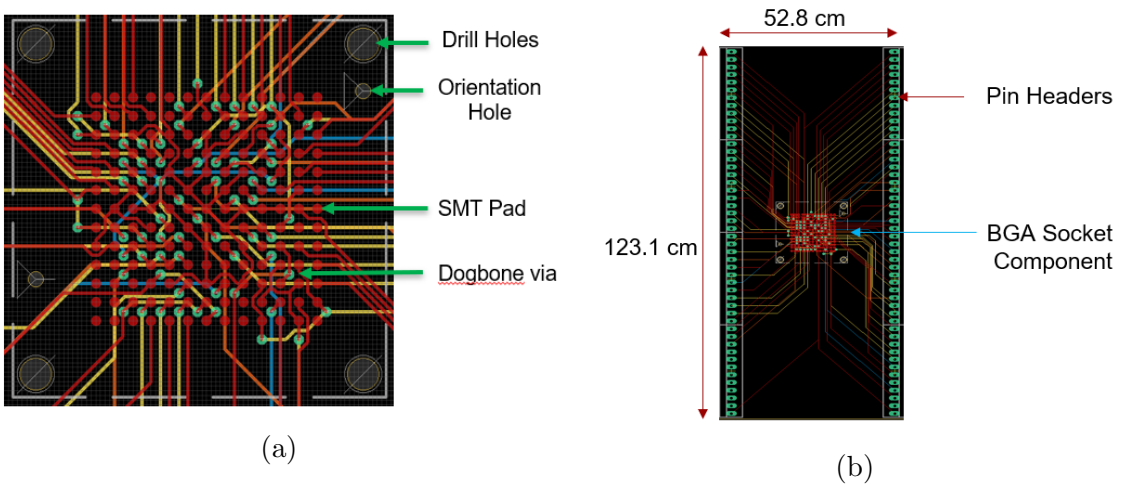


Figure 5.13: A BGA socket that allows testing multiple chips on a single PCB [16]

5.6 Testing

The designed PCB operates as a breakout board. Among the 169 balls of the chip, all functionally-equivalent balls were shorted in the PCB and then routed to 100 pin headers. The breakout board is designed to enable the chip to be mounted directly on a breadboard for testing.



(a) (b)

Figure 5.14: figure 5.14a shows the custom PCB created for the BGA socket. It shows the SMT pads corresponding to the BGA balls and the dogbone vias used for routing. Figure 5.14b shows the complete PCB along with pin headers.

When we detected the short between the peripheral power supply and ground on the packaged chip, we conducted various experiments to detect the location of the electrical short. First, a SPICE netlist was extracted along with parasitics from the layout used for tape-out and no electrical shorts were detected during the circuit simulation. We next check for metal to metal shorts in the power grids through ERC and LVS using Synopsys ICC. We introduced a short manually between the ground and power supplies to ensure that the ERC and LVS checks were being performed correctly. Although the manually shorted location was flagged by both ERC and LVS, no metal to metal shorts were detected in the original design. We were not able to perform LVS and ERC after full chip integration as the design hierarchy is removed at that stage. To ensure that the short was not introduced during packaging, the unpackaged die was tested using a probing-station. The probe station showed a low resistance connection between the peripheral power and ground pin indicating a short in the unpackaged die. Despite all the steps taken above, it is not clear where the electrical short occurred or how it was created.

CHAPTER 6

CONCLUSIONS

In this thesis, we have explored the effect of ramp-up times on the power-up SRAM PUF and observed the following:

- From SPICE simulation of SRAM cells in isolation, increasing power supply ramp time appears to increase the reliability of the power-up values.
- SPICE simulation of SRAM modules show that when the power supplies to the peripheral circuitry and the memory core are separated and sequenced, reliability increases with increasing ramp time of the memory core power supply. The reliability does not increase , however, if the supplies are not sequenced.
- From measurements made using commercial SRAM ICs, we noticed an apparent architecture-dependent trend in the reliability of the IC. For one of the tested SRAMs, we notice maximum reliability at a particular ramp time of 1 second, while for the other SRAM the reliability is largely not impacted by supply ramp times.
- We also observe a trend in the power-up values across commercial SRAMs where certain addresses tend to power-up to a particular state and the fraction of bits powering-up to state-1 tends toward 0.5 with increasing ramp time.

From these observations, we can conclude the following:

- Increasing ramp times may increase the reliability of the SRAM cells.

- Although more experiments and measurements are required, the preliminary results show that separating and sequencing the power supplies between the peripheral circuitry and the memory core could increase reliability and also influence power-up values.

- In commercial SRAM ICs, the architecture of the IC plays a role in determining the effect of ramp-up times on the SRAM.

- Ensuring the same ramp time between enrollment and key regeneration is important to ensure reliable power-up values.

- A ramp time can be chosen to either ensure high uniqueness or high reliability.

Apart from the above, we also provide a foundation for setting up a design flow that can be used to tape-out a SRAM with separated power supplies for the peripheral circuitry and memory core. Future work can involve testing the effect of supply ramp time on a custom taped-out silicon chip. Unlike a commercial chip, the knowledge of the SRAM architecture in the custom SRAM will help narrow down the impact of the peripheral circuitry on the power-up state of the SRAM. Moreover, a custom SRAM enables us individual control over the power delivery to the memory core and peripheral circuitry, which may shed light on the impact of the power supply sequencing on the power-up state of the SRAM.

BIBLIOGRAPHY

[1] Agarwal, Aseem, Blaauw, David, Zolotov, Vladimir, Sundareswaran, Savithiri, Zhao, Min, Gala, Kaushik, and Panda, Rajendran. Path-based statistical timing analysis considering inter-and intra-die correlations. In *Proceedings, TAU* (2002).

[2] Anderson, J. H. A PUF design for secure FPGA-based embedded systems. In *2010 15th Asia and South Pacific Design Automation Conference (ASP-DAC)* (Jan 2010), pp. 1–6.

[3] Anderson, Ross J. *Security Engineering: A Guide to Building Dependable Distributed Systems*, 2 ed. Wiley Publishing, 2008.

[4] Arduino. Arduino Mega 2560. `https://store.arduino.cc/usa/mega-2560-r3`, 2010.

[5] Asenov, A. Random dopant induced threshold voltage lowering and fluctuations in sub-0.1 /spl mu/m mosfet's: A 3-d "atomistic" simulation study. *IEEE Transactions on Electron Devices 45*, 12 (Dec 1998), 2505–2513.

[6] Bhargava, Mudit, and Mai, Ken. A High Reliability PUF Using Hot Carrier Injection Based Response Reinforcement. In *Cryptographic Hardware and Embedded Systems - CHES 2013* (Berlin, Heidelberg, 2013), Guido Bertoni and Jean-Sébastien Coron, Eds., Springer Berlin Heidelberg, pp. 90–106.

[7] Bösch, Christoph, Guajardo, Jorge, Sadeghi, Ahmad-Reza, Shokrollahi, Jamshid, and Tuyls, Pim. Efficient helper data key extractor on FPGAs. In *International Workshop on Cryptographic Hardware and Embedded Systems* (2008), Springer, pp. 181–197.

[8] Brzuska, Christina, Fischlin, Marc, Schröder, Heike, and Katzenbeisser, Stefan. Physically Uncloneable Functions in the universal composition framework. In *Annual Cryptology Conference* (2011), Springer, pp. 51–70.

[9] Chatterjee, U., Chakraborty, R. S., Mathew, J., and Pradhan, D. K. Memristor Based Arbiter PUF: Cryptanalysis Threat and Its Mitigation. In *2016 29th International Conference on VLSI Design and 2016 15th International Conference on Embedded Systems (VLSID)* (Jan 2016), pp. 535–540.

[10] Chen, Q., Csaba, G., Lugli, P., Schlichtmann, U., and Rhrmair, U. The Bistable Ring PUF: A new architecture for strong Physical Unclonable Functions. In *2011 IEEE International Symposium on Hardware-Oriented Security and Trust* (June 2011), pp. 134–141.

[11] Chen, Qingqing, Csaba, György, Lugli, Paolo, Schlichtmann, Ulf, and Rührmair, Ulrich. Characterization of the Bistable Ring PUF. In *Proceedings of the Conference on Design, Automation and Test in Europe* (San Jose, CA, USA, 2012), DATE '12, EDA Consortium, pp. 1459–1462.

[12] Delvaux, J., Gu, D., Schellekens, D., and Verbauwhede, I. Helper Data Algorithms for PUF-Based Key Generation: Overview and Analysis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 34*, 6 (June 2015), 889–902.

[13] Delvaux, Jeroen, Gu, Dawu, Schellekens, Dries, and Verbauwhede, Ingrid. Helper Data Algorithms for PUF-Based Key Generation: Overview and Analysis. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on 34* (06 2015), 889–902.

[14] Dodis, Yevgeniy, Reyzin, Leonid, and Smith, Adam. Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data. In *Advances in Cryptology - EUROCRYPT 2004* (Berlin, Heidelberg, 2004), Christian Cachin and Jan L. Camenisch, Eds., Springer Berlin Heidelberg, pp. 523–540.

[15] Drennan, P. G., and McAndrew, C. C. Understanding MOSFET mismatch for analog design. *IEEE Journal of Solid-State Circuits 38*, 3 (March 2003), 450–456.

[16] Electronics, Ironwood. . `https://www.ironwoodelectronics.com/catalog/Content/Templates/PartGrids.cfm?StartRow=161&cPart=SG-BGA-6455&Grid=SG-BGA_TABLE-1mm`.

[17] Garg, A., and Kim, T. T. Design of SRAM PUF with improved uniformity and reliability utilizing device aging effect. In *2014 IEEE International Symposium on Circuits and Systems (ISCAS)* (June 2014), pp. 1941–1944.

[18] Gassend, Blaise. Physical Random Functions. Master's thesis, Massachusetts Institute of Technology, MA, USA, 2003.

[19] Gassend, Blaise, Clarke, Dwaine, van Dijk, Marten, and Devadas, Srinivas. Silicon Physical Random Functions. In *Proceedings of the 9th ACM Conference on Computer and Communications Security* (New York, NY, USA, 2002), CCS '02, ACM, pp. 148–160.

[20] Ghosh, S., and Govindaraj, R. A strong arbiter PUF using resistive RAM. In *2016 International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation (SAMOS)* (July 2016), pp. 275–280.

[21] Guajardo, Jorge, Kumar, Sandeep S., Schrijen, Geert-Jan, and Tuyls, Pim. FPGA Intrinsic PUFs and Their Use for IP Protection. In *Cryptographic Hardware and Embedded Systems - CHES 2007* (Berlin, Heidelberg, 2007), Pascal Paillier and Ingrid Verbauwhede, Eds., Springer Berlin Heidelberg, pp. 63–80.

[22] H. Lundberg, Kent. Noise Sources in Bulk CMOS.

[23] Holcomb, D. E., Burleson, W. P., and Fu, K. Power-Up SRAM State as an Identifying Fingerprint and Source of True Random Numbers. *IEEE Transactions on Computers 58*, 9 (Sep. 2009), 1198–1210.

[24] Holcomb, Daniel E, Burleson, Wayne P, Fu, Kevin, et al. Initial SRAM state as a fingerprint and source of true random numbers for RFID tags. In *Proceedings of the Conference on RFID Security* (2007), vol. 7, p. 01.

[25] Holdings, ARM. ARM Artisan Memory Compiler 16nm FinFET. `https://developer.arm.com/ip-products/physical-ip/embedded-memory`, 2010.

[26] Instruments, Texas. LMC6482. `http://www.ti.com/lit/ds/symlink/lmc6482.pdf`, 2015.

[27] Islam, M. N., Patil, V. C., and Kundu, S. On Enhancing Reliability of Weak PUFs via Intelligent Post-Silicon Accelerated Aging. *IEEE Transactions on Circuits and Systems I: Regular Papers 65*, 3 (March 2018), 960–969.

[28] Jang, J., and Ghosh, S. Design and analysis of novel SRAM PUFs with embedded latch for robustness. In *Sixteenth International Symposium on Quality Electronic Design* (March 2015), pp. 298–302.

[29] Kim, T. W., Choi, B. D., and Kim, D. K. Zero bit error rate ID generation circuit using via formation probability in 0.18 m CMOS process. *Electronics Letters 50*, 12 (June 2014), 876–877.

[30] Kumar, S. S., Guajardo, J., Maes, R., Schrijen, G., and Tuyls, P. Extended abstract: The butterfly puf protecting ip on every fpga. In *2008 IEEE International Workshop on Hardware-Oriented Security and Trust* (June 2008), pp. 67–70.

[31] Maes, R., Tuyls, P., and Verbauwhede, I. A soft decision helper data algorithm for SRAM PUFs. In *2009 IEEE International Symposium on Information Theory* (June 2009), pp. 2101–2105.

[32] Maes, Roel, and Verbauwhede, Ingrid. *Physically Unclonable Functions: A Study on the State of the Art and Future Research Directions*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 3–37.

[33] Memory, Alliance. AS6C6264. `https://www.alliancememory.com/wp-content/uploads/pdf/Alliance%20Memory_64K_AS6C6264v2.0July2017.pdf`, 2017.

[34] Menhorn, Nathan. External Secure Storage Using the PUF. `https://www.xilinx.com/support/documentation/application_notes/xapp1333-external-storage-puf.pdf`, 2018.

[35] Merli, Dominik, Schuster, Dieter, Stumpf, Frederic, and Sigl, Georg. Side-Channel Analysis of PUFs and Fuzzy Extractors. In *Trust and Trustworthy Computing* (Berlin, Heidelberg, 2011), Jonathan M. McCune, Boris Balacheff, Adrian Perrig, Ahmad-Reza Sadeghi, Angela Sasse, and Yolanta Beres, Eds., Springer Berlin Heidelberg, pp. 33–47.

[36] Nassif, S., Bernstein, K., Frank, D. J., Gattiker, A., Haensch, W., Ji, B. L., Nowak, E., Pearson, D., and Rohrer, N. J. High Performance CMOS Variability in the 65nm Regime and Beyond. In *2007 IEEE International Electron Devices Meeting* (Dec 2007), pp. 569–571.

[37] Pappu, Ravikanth, Recht, Ben, Taylor, Jason, and Gershenfeld, Neil. Physical One-Way Functions. *Science 297*, 5589 (2002), 2026–2030.

[38] Paral, Z., and Devadas, S. Reliable and efficient PUF-based key generation using pattern matching. In *2011 IEEE International Symposium on Hardware-Oriented Security and Trust* (June 2011), pp. 128–133.

[39] Patil, V. C., Vijayakumar, A., Holcomb, D. E., and Kundu, S. Improving reliability of weak PUFs via circuit techniques to enhance mismatch. In *2017 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)* (May 2017), pp. 146–150.

[40] Ravikanth, Pappu. *Physical One-Way Functions*. PhD thesis, Massachusetts Institute of Technology, 3 2001.

[41] Rührmair, Ulrich, Busch, Heike, and Katzenbeisser, Stefan. *Strong PUFs: Models, Constructions, and Security Proofs*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 79–96.

[42] Rührmair, Ulrich, and Holcomb, Daniel E. PUFs at a Glance. In *Proceedings of the Conference on Design, Automation & Test in Europe* (3001 Leuven, Belgium, Belgium, 2014), DATE '14, European Design and Automation Association, pp. 347:1–347:6.

[43] Shyu, J. ., Temes, G. C., and Krummenacher, F. Random error effects in matched MOS capacitors and current sources. *IEEE Journal of Solid-State Circuits 19*, 6 (Dec 1984), 948–956.

[44] Spenke, Alexander, Breithaupt, Ralph, and Plaga, Rainer. An arbiter PUF secured by remote random reconfigurations of an FPGA. *CoRR abs/1610.04065* (2016).

[45] Suh, G. E., and Devadas, S. Physical Unclonable Functions for Device Authentication and Secret Key Generation. In *2007 44th ACM/IEEE Design Automation Conference* (June 2007), pp. 9–14.

[46] Suh, G. E., O'Donnell, C. W., and Devadas, S. Aegis: A single-chip secure processor. *IEEE Design Test of Computers 24*, 6 (Nov 2007), 570–580.

[47] Suh, G. Edward, and Devadas, Srinivas. Physical Unclonable Functions for Device Authentication and Secret Key Generation. In *Proceedings of the 44th Annual Design Automation Conference* (New York, NY, USA, 2007), DAC '07, ACM, pp. 9–14.

[48] Technologies, Keysight. 33511B Waveform Generator with Arb. https://www.keysight.com/en/pd-2155031-pn-33511B/waveform-generator-20-mhz-1-channel-with-arb?nid=-536902257.1026947&cc=US&lc=eng, 2010.

[49] Technologies, Keysight. 33503A Benchlink Waveform Builder Pro Software. https://www.keysight.com/en/pd-1962285-pn-33503A/benchlink-waveform-builder-pro-software?cc=US&lc=eng, 2017.

[50] Technologies, Microchip. 23LC1024 SRAM IC. https://www.microchip.com/wwwproducts/en/23LC1024, 2012-2015.

[51] Tu-Hsiung Tsai, Hung-Ming Chen, Hung-Chun Li Shi-Hao Chen. An efficient RDL routing for flip-chip designs. https://www.edn.com/design/systems-design/4419930/An-efficient-RDL-routing-for-flip-chip-designs, 2013.

[52] Tuyls, Pim, Schrijen, Geert-Jan, Škorić, Boris, van Geloven, Jan, Verhaegh, Nynke, and Wolters, Rob. Read-Proof Hardware from Protective Coatings. In *Cryptographic Hardware and Embedded Systems - CHES 2006* (Berlin, Heidelberg, 2006), Louis Goubin and Mitsuru Matsui, Eds., Springer Berlin Heidelberg, pp. 369–383.

[53] Usmani, M. A., Keshavarz, S., Matthews, E., Shannon, L., Tessier, R., and Holcomb, D. E. Efficient PUF-Based Key Generation in FPGAs Using Per-Device Configuration. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems 27*, 2 (Feb 2019), 364–375.

[54] Weste, Neil, and Harris, David. *CMOS VLSI Design: A Circuits and Systems Perspective*, 4th ed. Addison-Wesley Publishing Company, USA, 2010.

[55] Willers, Oliver, Huth, Christopher, Guajardo, Jorge, and Seidel, Helmut. MEMS Gyroscopes As Physical Unclonable Functions. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (New York, NY, USA, 2016), CCS '16, ACM, pp. 591–602.

[56] Yu, M., and Devadas, S. Secure and robust error correction for Physical Unclonable Functions. *IEEE Design Test of Computers 27*, 1 (Jan 2010), 48–65.